מחשבת

בצאת ישראל ממצרים

ATARI 10

600XL 800XL ATARI 400and 800

ATARI
600XL, 800XL, 400, 800

מחשבת

בצאת ישראל ממצרים

יחידה 10
משלבים כוחות
ליצירת משחק פעולה

לחץ כאן כדי לשחק
או להוריד את המשחק

תוכן העניינים

הקדמת המחבר

כתיבת שלושת החוברות: מחשבת 7, מחשבת 8 ומחשבת 9, הייתה עבורי מסע. מסע אשר במהלכו צברתי כלים תכנותיים רבים בכלל, והעמקתי את הידע ברזי מחשב האטארי, בפרט.

בנקודה זו החלטתי לרתום את הידע שצברתי ולתרגם אותו לבניית משחק פעולה. מבנה המשחק שהחלטתי לכתוב הוא של: Space Invaders, ומבחינה זו "לא המצאתי את הגלגל". אולם כדי לחדש ולהוסיף למשחק אופי ישראלי, אימצתי את סיפור יציאת מצרים: את ה"פולשים" בניתי כפרעונים, את דמות השחקן – כאחת ממכות מצרים, ואת שכבת ההגנה – כשורת פירמידות. הכיתוב המלווה את המשחק הוא כמובן בעברית.

סביבת התכנות שבה עבדתי היא כרגיל – הבייסיק, אך נעזרתי בכמה וכמה רוטינות אסמבלי. המשחק נכתב בעזרת האמולטור ALTIRRA; הוא נבדק ופועל על מחשב ATARI 800XL, אם כי באיטיות! על מנת להפעילו על מחשב האטארי מומלץ **לא** להשתמש בבייסיק המקורי, אלא ב-TURBO-BASIC.

עם סיום כתיבת המשחק, ראיתי לנכון לחלוק ולשתף את הידע. לשם כך, העליתי על הכתב את שורות הקוד, כשהן מלוות בהסברים. במקומות מסוימים הרחבתי את היריעה וסקרתי את העקרונות התכנותיים שהנחו אותי, ואף התעכבתי על כמה פקודות בייסיק ואסמבלי – פקודות אשר טרם הוזכרו באף אחת מ-9 חוברות המחשבת הקיימות.

למותר לציין שמלאכת התכנות בכלל, וכתיבת משחק פעולה בפרט – בדומה לכתיבת סיפור – היא תהליך יצירתי. מבחינה זו, סביר בהחלט שהדרכים בהן בניתי את מקטעי הקוד שלי יכלו להיכתב גם אחרת – ייתכן אף באופן יעיל יותר. הקוראים מוזמנים להציע רעיונות ושיפורים.

נדב שטכמן פולישוק
אפריל 2022

מבוא

המשחק "מכות מצרים" מורכב מארבעה קבצי בייסיק נפרדים:

1. HEBREW.BAS – קובץ המתקין את האותיות העבריות.
2. SEDER.BAS – קובץ הטוען לזיכרון את רוטינות האסמבלי המשתלבות בתוכנת הבייסיק. קובץ זה אחראי גם למסך הפתיחה של המשחק "מכות מצרים".
3. GAME.BAS – המשחק עצמו.
4. WINNER.BAS – קובץ המטפל בתמונת הניצחון עם סיום כל משימות המשחק.

בפרקים שלהלן יוצגו המשימות התכנותיות העיקריות שנדרשו לי לשם בניית המשחק. אסביר כיצד יצרתי את האלמנטים הגרפיים השונים על המסך; כיצד טיפלתי בתנועת השחקנים, האויבים והטילים ובכללי המפגש ביניהם; באיזה טכניקה נעזרתי לבניית מוסיקת הרקע; ולבסוף, באילו בעיות ובאגים נתקלתי במהלך התכנות וכיצד התמודדתי עמם.

אדגיש כי החוברת אינה מתעכבת על כל שורת קוד, וזאת מתוך הנחה שהקוראים מכירים לעומקה את שפת הבייסיק ושולטים ביסודות התכנות באסמבלי. על מנת להפיק מן החוברת את המיטב, רצוי לקרוא באופן יסודי את כל סדרת החוברות הקודמות ובעיקר את מחשבת 6, 7, 8, 9, ובלשון ההגדה של פסח – כל המרבה להעמיק בלימוד האטארי הרי זה משובח!

ולבסוף, הערה טכנית. את החוברת "מחשבת 10" כתבתי בצורה מעט שונה, במובן זה שאין בה את מבנה השאלות והתשובות שליוו את החוברות הקודמות. יחד עם זאת, היא מתכתבת עם קודמותיה בסגנון הכתיבה, ובצורת הצגת הדברים, ומפנה אליהן במקרי הצורך.

פרק א
הכנות מקדימות

1. הקצאת זיכרון

לפני שיוצאים לדרך ופותחים במלאכת התכנות, מומלץ מאוד לתכנן את "מגרש המשחקים" שלנו.

אנו עובדים במחשב ATARI 800XL, בסביבת בייסיק, ותחת ניהול כונן דיסקטים. לכן כמות הזיכרון הזמינה לנו לעבודה היא כ-35 קילובייט.

המשחק שנבנה כולל שימוש בגרפיקת השחקנים (PM), שימוש בעברית ובתווי טקסט ייחודיים וכמובן ניעזר ברוטינות שפת מכונה.

להלן מפת הזיכרון שתלווה אותנו לאורך כתיבת המשחק:

ROM
עברית +תווי עזר
זיכרון PM
DATA ותוכנות עזר נוספות
זיכרון מסך
D.List
תוכנות עזר דף 6

160
156
152
150
149
$8C55
35925
6

א. את האותיות העברית (כמו גם תווים מיוחדים שישמשו אותנו לעיצוב הדמויות) נמקם 1K מתחת ל-ROM, כלומר בארבעת הדפים החל מדף 156.[1]

ב. עבור זיכרון גרפיקת השחקנים נקצה את ארבעת הדפים החל מדף 152.

ג. לרוטינות שפת המכונה נקצה את דף 6, אך מכיוון שמדובר בכמה וכמה רוטינות שכאלה, נפנה לעצמנו 2 דפים נוספים, החל מדף 150.

ד. לבסוף, כאשר אנחנו מורידים את ה-RAMTOP עלינו להביא בחשבון כי הזיכרון אשר מקצה הבייסיק לטיפול במסכים הגרפיים השונים עשוי לגרום לתקלות.

כזכור ממחשבת 9, הארכיטקטורה של מחשב האטארי לא מאפשרת לטפל בעת ובעונה אחת בלמעלה מ-1K של זיכרון. יתרה על כן, זיכרון המסך חייב לשבת על כפולה שלמה של 1K, אלא אם נוסיף פקודות דילוג בתוך ה-Display List (כפי שאכן קורה במסך גרפי 15 למשל).

על מנת למקם את זיכרון המסך על כפולה שלמה של 1K ולהימנע מפקודות דילוג, נוריד את ה-RAMTOP בעוד 256 בתים, כלומר בסך הכול עד דף 149.

2. עברית במהירות הבזק

בסיומה של מחשבת 6 למדנו לעצב את מערך סימני העברית. לשם כך, העתקנו תחילה את מקטע זיכרון התווים מן ה-ROM ל-RAM, ואחר כך הכנסנו את ערכי התווים אשר עיצבנו למקומם הנדרש בזיכרון בעזרת פקודות הבייסיק: **READ** ו-**DATA**.

1 כזכור, דף הוא יחידה של 256 בתים. 1K הם 4 דפים, או 1024 בתים.

שתי משימות אלו גוזלות זמן רב יחסית (למעלה מ-5 שניות), וכבר נוכחנו במחשבת 8 שאפשר לקצר משמעותית את משך זמן המשימה לכמה חלקיקי שנייה, זאת אם נכתוב את הקוד באסמבלי.

דא עקה, אם נרצה להוסיף את רוטינת שפת המכונה שכתבנו, לתוך תוכנת בייסיק, נידרש להכניס (לתוך דף 6 למשל), גם את כל ערכי התווים שעיצבנו (27 תווים בגודל 8 סיביות כל אחד, כלומר בסך הכול 216 ערכים), שוב בעזרת פקודות הבייסיק: **READ** ו-**DATA**. וכך יצא שכרנו בהפסדנו!

? האם יש שיטה מהירה יותר לבצע את המשימה?

התשובה היא: כן.

לשם כך נוכל להיעזר בתא מחרוזתי בן 216 מקומות, לאכסן בתוכו את כל ערכי התווים שעיצבנו (פעולה מהירה ביותר אפילו בבייסיק), ולהעתיק את המידע הזה למקום הרצוי בזיכרון, במהירות הבזק, באמצעות רוטינת אסמבלי קצרה.

? כיצד נעתיק את המידע מן התא המחרוזתי אל המקום הרצוי לנו בזיכרון?

באמצעות שימוש בפקודת הבייסיק **ADR** ושליחת כתובת הזיכרון **הדינאמי** שהיא מספקת לנו אל רוטינת אסמבלי קצרה שנבנה.

אם אינכם זוכרים כיצד להשתמש בפקודת הבייסיק **ADR**, תוכלו לרענן את זיכרונכם ולעיין בפרק "מגבירים את הקצב" במחשבת 7, ואם אינכם זוכרים כיצד שולחים פרמטרים מתוכנת הבייסיק אל רוטינה בשפת מכונה, חזרו ועיינו בסעיף "גלילה עדינה" במחשבת 9 – העוסק בשימוש מורכב בפקודת הבייסיק **USR**.

נסכם את שלבי העבודה הנדרשים:

1. עלינו לאכסן את 216 הערכים המרכיבים את 27 האותיות העבריות שעיצבנו בתוך תא מחרוזתי.
2. עלינו להשיג את כתובת הזיכרון **הדינאמית** בה שמור אותו תא מחרוזתי, באמצעות פקודת הבייסיק **ADR**.
3. עלינו לבנות רוטינת אסמבלי המבצעת שתי משימות:

א. העתקת מערך הסימנים הקיים מן ה-ROM ל-RAM (כפי שכבר עשינו בסעיף: "אמור גם בעברית שלום" במחשבת 8).

ב. העתקת מקטע הזיכרון הדינאמי מן הכתובת שתישלח לרוטינה, אל כתובת הזיכרון הקבועה אשר בה יישבו בסופו של דבר האותיות העברית. לשם כך ניעזר בפקודת הבייסיק **USR** עם פרמטר.

לפני שניגש למשימת התכנות אצביע על שתי בעיות – הראשונה טכנית, והשנייה תכנותית.

ראשית, על מנת להכניס את נתוני האותיות העבריות שעיצבנו לתוך תא מחרוזתי, נצטרך להקליד בתוכנית הבייסיק שלנו, רצף של 216 תווים (שהרי בתא מחרוזתי איננו שומרים נתונים מספריים, אלא את ייצוגם הגרפי). ואולם חלק מערכי ה-ASCII מתורגמים לתווים מיוחדים (כגון ארבעת החצים, תו ה-BACKSPACE, וכולי).

על מנת להקליד את התווים הללו, נוכל להיעזר בבית 766. אם נפקוד לתוכו את הערך 1, יוצגו כל אותם התווים המיוחדים כתווים "רגילים". כך למשל תו ה-BACKSPACE יוצג על המסך כ-"◄". על מנת להשיב את המצב לקדמותו נפקוד לתוך בית 766 את הערך 0.

שנית, עלינו למצוא דרך להפוך את הפרמטר שמעובר מן הבייסיק אל רוטינת שפת המכונה לכתובת **שלמה אחת**. נניח לצורך הדוגמה כי כתובת הזיכרון של התא המחרוזתי שלנו היא 7618, וכי הרוטינה שלנו יושבת בכתובת 1536.

כאשר נקרא לרוטינה שלנו, כך:

```
U=USR(1536,7618)
```

יישמרו בתוך המחסנית הערכים: 29 ו-194 (לפי שיטת ה-MSD, LSD).

? כיצד נוכל להפוך את שני הערכים הללו **בתוך רוטינת שפת המכונה**, בחזרה לכתובת שלמה אחת?

כאן בא לעזרתו שימוש שטרם היכרנו בפקודת האסמבלי **LDA**.

כפי שנראה מיד, נוכל **לשלוח** את האוגר אל כתובת זיכרון מסוימת, המורכבת משני תאי זיכרון **עוקבים**, ולפקוד עליו לקחת את התוכן השמור **בכתובת** הזו.

אדגים.

נניח שבתא זיכרון 203, שמור הערך: 64, בתא הזיכרון העוקב, 204 שמור הערך: 156, וברגיסטר Y שמור הערך 1.

אם נפקוד:

```
LDA  ($CB),Y
```

האוגר ייגש אל תא הזיכרון שכתובתו 40001 וייקח את הערך השמור בו.

? מדוע כתובת 40001?

משום שהאוגר מתייחס לבתים 203 ו-204 (או $CB ו-$CC) כאל זוג בתים מצביעים, ומרכיב באמצעותם לפי שיטת ה-LSD-MSD, את הכתובת:

156*256 + 64 = 40000

ואת הכתובת הזאת מקדם בהתאם לערכו של רגיסטר Y – שהוא 1.

שימו לב:

1. בשימוש המיוחד הזה בפקודה **LDA** – המכונה **"מיעון עקיף, Y"** יש לשים בסוגריים את התא המצביע על החלק הפחות משמעותי (LSD) של הכתובת, מתוך הנחה, כמובן, שבתא העוקב, נמצא החלק היותר משמעותי (MSD) של הכתובת.
2. **חייבים** להשתמש בנוסף לאוגר, גם ברגיסטר Y, **ובו בלבד**. כמובן שרגיסטר Y יכול לשמש כמונה שאיתו נתקדם לאורך תאי הזיכרון.
3. שני תאי הזיכרון המשמשים כבתים מצביעים **חייבים** לשבת בתוך דף מספר 0 (כלומר בטווח התאים שבין 0 ל-255)

   דעו לכם כי בתוך דף מספר 0 קשה מאוד למצוא שני תאי זיכרון עוקבים שאין להם כל שימוש – לא של מערכת ההפעלה ולא של סביבת הבייסיק. שני התאים 203, 204 הם בין התאים הבודדים הפנויים בדף מספר 0, אשר נוכל לעשות בהם שימוש ללא חשש.
4. אל תוך הבית הנמוך מבין השניים (203 או $CB) נכניס את החלק הפחות משמעותי (LSD) של הכתובת; אל תוך הבית הגבוה מבין השניים (204 או $CC) נכניס את החלק היותר משמעותי (MSD) של הכתובת, ולסוגריים שבפקודה **LDA** נכניס את הבית **הנמוך** בלבד.

עתה אגש לתכנות עצמו, ואציג את רוטינת האסמבלי לביצוע המשימה:

```
10    *=$600
20    LDY #$0
30    PLA
40    PLA
50    STA $CC
60    PLA
70    STA $CB
80    LDX #$0
90  LOOP
100    LDA $E000,X
110    STA $9C00,X
120    LDA $E100,X
130    STA $9D00,X
140    LDA $E200,X
150    STA $9E00,X
160    LDA $E300,X
170    STA $9F00,X
180    CPX #$FF
190    BEQ HEB
200    INX
210    JMP LOOP
220 HEB
230    INY
240    CPY #$D8
250    BEQ EXIT
260    LDA ($CB),Y
270    STA $9F07,Y
280    JMP HEB
290 EXIT
300    RTS
```

הסבר:

בשורות 70-20 נשלפים מן המחסנית ערכי הפרמטר אשר נשלחו אל הרוטינה מתוכנת הבייסיק והם מאוכסנים בשני הבתים העוקבים: 203, 204.

בשורות 210-90 מועתק מערך התווים מן ה-ROM אל ה-RAM.

בשורות 280-220 מועתק מקטע הזיכרון **הדינאמי** (בו יושב המערך המחרוזתי המכיל את מידע האותיות בעברית) אל המקום הקבוע המיועד לו.

שימו לב לשורה 260, בה נעשה השימוש שזה אך למדנו בפקודה: **LDA**.

בחזרה אל הבייסיק

בנוסף לחוברת זו מוצגת תוכנת הבייסיק: HEBREW.BAS אשר טוענת את מערך הסימנים כולו – לרבות אותיות העברית המעוצבות – אל מקומם הייעודי ב-RAM.

בשורות הבייסיק 50-30 מאוכסן המידע המעצב את אותיות העברית בתא מחרוזתי בשם: HEB$. שורות הבייסיק 130-100 ו-1020-1000 טוענות את הרוטינה שלעיל (המתורגמת לשפת מכונה) אל דף מספר 6. ושורה 210 מריצה את הרוטינה על הפרמטר שנשלח אליה – הלוא היא כתובת הזיכרון של המשתנה המחרוזתי HEB$.

שורות 230-220 אינן קשורות למשימת עיצוב אותיות העברית, אלא למשחק "מכות מצרים" גופא, והן הוכנסו אל תוך הקובץ HEBREW.BAS משיקולי חיסכון בזיכרון.

3. תמונת הפתיחה

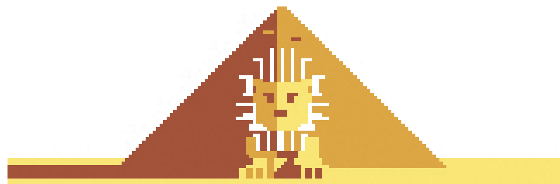
הרזולוציה המקסימלית שבה ניתן להציג תמונות במחשב האטארי היא, כידוע, 160X160 פיקסלים, ומספר הצבעים המקסימלי העומד לרשותנו הוא 3 (זאת בנוסף לצבע הרקע).[2]

להלן אציג את האופן שבו הפכתי את תמונת הספינקס שעל רקע הפירמידה, לתמונה הניתנת להצגה במחשב האטארי. לצורך כך נעזרתי בתוכנה פוטושופ של אדובי.[3]



בשלב הראשון, דאגתי שרזולוציית התמונה תהייה: 160X54 פיקסלים.

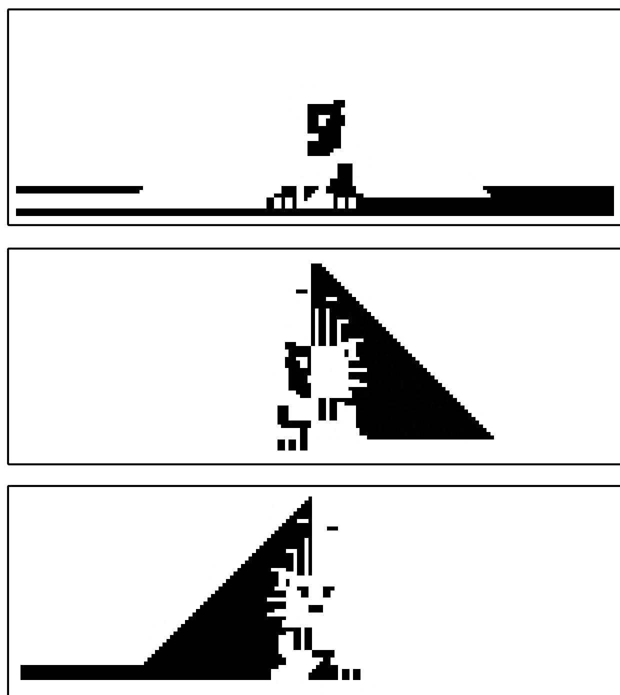
בנוסף, וידאתי כי מספר הצבעים שנעשה בהם שימוש הוא 3. את הרקע הכחול של השמיים הסרתי מן התמונה, כמו גם את הרעמה של האריה (הצבע החמישי), מתוך כוונה להוסיף פריט זה בהמשך (באמצעות שימוש בגרפיקת השחקנים). זו תוצאת הביניים שהתקבלה:



2 בנוסף, אפשר כמובן לנצל את יכולותיו של ה-GTIA וליהנות ממספר גבוה יותר של צבעים.

3 ניתן כמובן להשתמש בכל תוכנה גרפית אחרת, אשר מאפשרת לשלוט במספר הפיקסלים לאורך ולרוחב, כמו גם במספר הצבעים בתמונה.

את התמונה שנותרה, הפרדתי ל-3 משטחים – אחד עבור הצבע הצהוב, אחד עבור הצבע הכתום ואחד עבור הצבע החום:



את שלושת משטחי הצבע המרתי לקובצי טקסט בינאריים באמצעות אתר האינטרנט: Image to Binary Converter (כל פיקסל שחור מתורגם ל-1, וכל פיקסל לבן מתורגם ל-0), ואת קבצי הטקסט הללו העברתי לדיסקט אטארי (בעזרת אפליקציה בשם: AtrUtil).

בשלב זה כתבתי תוכנת בייסיק קצרה, שקוראת כל אחד מקבצי הטקסט, ומציירת (באמצעות הפקודה: **PLOT** נקודה על המסך הגרפי בכל אימת שנקרא הערך 1 מתוך הקובץ. בין קריאת קובץ אחד למשנהו החלפתי כמובן את צבע ה"עיפרון" (באמצעות הפקודה: **COLOR**). הגרפיקה שהשתמשתי בה היא גרפיקה 7 בבייסיק.

לבסוף, פקדתי על המחשב לסרוק את אזור זיכרון המסך מתחילתו ועד סופו ושמרתי את המידע בתוך תא מחרוזתי גדול דיו.

מסך הפתיחה מחולק למעשה לשניים – בחלק העליון מתנוסס שם המשחק ושם כותבו (עבדכם הנאמן), ואילו בחלק התחתון מופיע האיור. לשם כך, נעזרתי בהגדרות המובנות של מסך גרפי מספר 2, אך עשיתי מניפולציה על ה-Display List שלו:

בראש המסך הותרתי 3 יחידות מסוג גרפיקה 2, ויחידה אחת מסוג גרפיקה 1. בתחתית המסך הותרתי חלון טקסט בן ארבע יחידות (מסוג גרפיקה 0), ואילו בתווך קבעתי 51 יחידות מסוג גרפיקה 7.

אם נסכם את מספר קווי הסריקה, נקבל: 16X3 + 8X1 + 8X4 + 2X51 ובסך הכול 190 קווי סריקה.

לגבי זיכרון המסך של אזור האיור – קרי אותן 51 יחידות מסוג גרפיקה 7 – במקום להקצות עבורן זיכרון קבוע, ניתבתי את ה-Display List לכתובת **הזיכרון הדינאמי** של התא המחרוזתי המאכסן את נתוני האיור.

את הכותבת הדינאמית הזו, קיבלתי כמובן, באמצעות שימוש בפקודת הבייסיק: **ADR**.

בנוסף לחוברת זו מוצגת תוכנת הבייסיק: SEDER.BAS אשר אחראית, בין היתר, לבניית מסך הפתיחה למשחק.

שורות הבייסיק 130-570 טוענות את נתוני האיור אל תוך תא מחרוזתי בשם A$. מכיוון שארכיטקטורת ה-Antic (המעבד הגרפי של האטארי) אינה מאפשרת כאמור, טיפול בטווח תאי זיכרון החורג מקילובייט שלם אחד, דאגתי שהמחשב ימקם את אותו תא מחרוזתי על כפולה **שלמה** של 1024 בתים.

4. עיצוב הפרעונים

שני המסכים הגרפיים 12 ו-13 מתאימים במיוחד לבניית משחקי פעולה. מסכים אלו מאפשרים הן שמירה על רזולוציה גבוהה, והן שימוש בארבעה צבעים, וכל זאת אגב הקצאת כמות נמוכה יחסית של זיכרון.

דעו לכם כי מסכים 12 ו-13 הם למעשה מסכים טקסטואליים, וככאלו ניתן לכתוב בהם טקסט באמצעות הפקודה: **PRINT #6**

נסו להקליד במסך 12 או 13 את הפקודה:

```
PRINT #6; "Hello"
```

התוצאה המתקבלת אינה מרשימה במיוחד. במקום אותיות ברורות, מופיעים כתמי צבע, אשר מזכירים את האותיות הלטיניות אך במעט.

? האם יש דרך לשלוט באותם כתמי צבע ולייצר אותיות ברורות וצבעוניות?

התשובה היא כן, אך לשם כך עלינו להבין את הלוגיקה של עיצוב האותיות במסכים גרפיים אלו.

נתבונן על התו: **A**. כידוע, נדרשים 8 בתים, שהם 64 סיביות על מנת לאכסן את המידע המייצר את אותו תו, ונוכל לשרטט זאת במטריצה הבאה:

			1	1			
		1	1	1	1		
	1	1			1	1	
	1	1			1	1	
	1	1	1	1	1	1	
	1	1			1	1	

עתה, **נניח** שבמקום לטעון במטריצה את הערך 0 (סיבית כבויה) או את הערך 1 (סיבית דולקת), יכולנו לטעון כל אחד מן הערכים: 0, 1, 2 או 3.

במקרה שכזה, יכולנו לצבוע את כל התו (או אזורים מסוימים בו) באחד משלושה צבעים. לדוגמה:

128	64	32	16	8	4	2	1
			1	1			
		1	1	1	1		
	2	2			2	2	
	2	2			2	2	
	3	3	3	3	3	3	
	3	3			3	3	

? האם ניתן לטעון סיבית בערך שונה מערך בינארי?

התשובה היא: בוודאי שלא!

אך בהחלט ניתן לגייס **מספר כפול של סיביות** לטיפול במשימת עיצוב האותיות. כלומר, ניתן להגדיר שכל שתי סיביות תתפקדנה כ"פיקסל" יחיד בתוך האות.

שיטה זו היא כמובן בזבזנית, משום שהיא דורשת במקום 64 סיביות, 128 סיביות ליצירת אות בודדת אחת. ואולם, מכיוון ששתי סיביות יכולות לשאת יחדיו , אחד מארבעה ערכים (0, 1, 2, או 3) – נוכל ליהנות מאותיות צבעוניות.

נחזור לדוגמת האות **A**, ונצייר אותה במטריצה ה"בזבזנית":

128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
						0	1	0	1						
				0	1	0	1	0	1	0	1				
		1	0	1	0					1	0	1	0		
		1	0	1	0					1	0	1	0		
		1	1	1	1	1	1	1	1	1	1	1	1		
		1	1	1	1					1	1	1	1		

שימו לב, מספר השורות לא השתנה, והוא נותר 8, אך בכל שורה אנו משתמשים במספר כפול של סיביות (16 סיביות, במקום 8). כלומר, על מנת לעצב כל אות, נזדקק **לשני תווים** שירכיבו יחידו את אותה אות.

? כיצד נחשב את הערכים אשר יקבלו שני התווים המרכיבים את האות הצבעונית A?

ובכן כפי שניתן לראות באיור, כל זוג סיביות מקבל בשיטה הבינארית את אחד הערכים: 01, 10 או 11. הצמד 01 ייצג צבע ראשון, הצמד 10 ייצג צבע שני, הצמד 11 ייצג צבע שלישי, ואילו הצמד 00 יותיר את ה"פיקסל" כבוי.

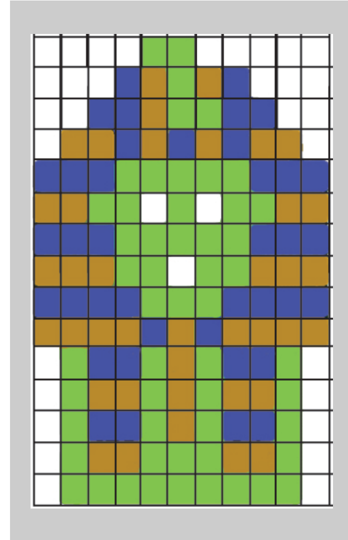
נחבר, לפי "שיטת המשקולות" בכל שורה את כל הסיביות הדולקות ונקבל:

0, 64, 80, 40, 252, 60, 0

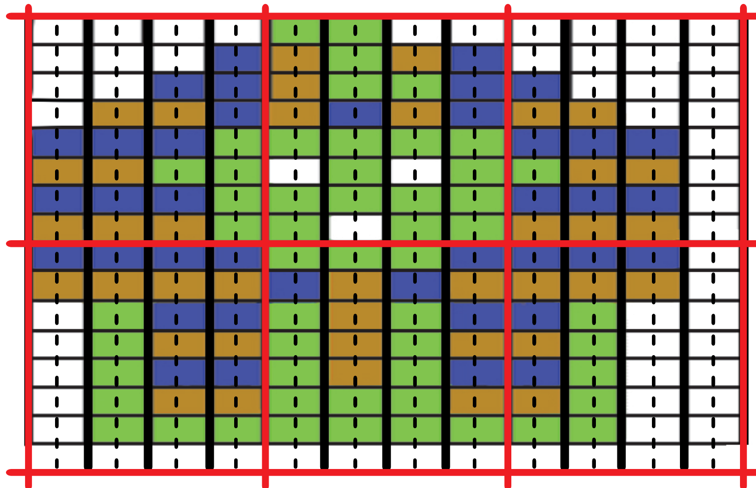
0, 1, 5, 40, 40, 63, 60, 0

בדיוק באותו אופן, בניתי את דמות הפרעון.

ראשית, שרטטתי את הדמות על מטריצה של 16X12:



אחר כך, חילקתי כל "פיקסל" לשתי סיביות (לפי "השיטה הבזבזנית"), כך שדמות הפרעון נמתחה והתפרסה על שישה תווים, כפי שמציג השרטוט הבא:



אלו הם נתוני ששת התווים היוצרים יחדיו את דמות הפרעון (משמאל לימין, ומלמעלה למטה):

0, 3, 15, 23, 254, 90, 254, 86

160, 103, 107, 119, 170, 34, 170, 138

0, 0, 192, 80, 252, 148, 252, 84

255, 85, 47, 37, 47, 37, 42, 0

171, 221, 155, 153, 155, 169, 170, 0

252, 84, 224, 96, 224, 96, 160, 0

5. מוסיקת רקע

במחשבת 9 למדנו לרתום את הפרעת ה-VBI להשמעת מוסיקת רקע (מוסיקה המנוגנת שוב ושוב בלי הפסקה וללא תלות בתוכנה). למעשה, הפרעת ה-VBI שימשה עבורנו מעין לולאה אינסופית ואל תוך לולאה זו שרבבנו רוטינה קטנה משלנו המנגנת את צלילי המוסיקה.

אלא שבפיתוח ובכתיבה של משחקי פעולה, אנו עשויים להידרש ליותר מלולאה אינסופית אחת. כך למשל, אם ברצוננו להניע באופן מחזורי ובלתי תלוי אלמנטים גרפיים מסוימים על המסך, יהיה מועיל ביותר להקצות עבור משימה זו לולאה אחת, ועבור מוסיקת הרקע לולאה אחרת.

בסעיף זה אציג לולאה מובנית הקיימת במחשב אטארי וניתנת לשימוש אך ורק מתוך רוטינה של שפת מכונה. לולאה זו מתבססת אף היא על הפעולה המחזורית של ריענון המסך, אך השימוש בה שונה מן השימוש בהפרעת VBI.

השימוש בלולאה המובנית הזו דורש כתיבה של שני מקטעי קוד: ראשית, ציון מיקומה של הפעולה המחזורית וקביעת תדירות חזרתה; ושנית, תכנות הפעולה המחזורית גופא. שני מקטעים אלו קשורים זה בזה וחייבים להיות גם

צמודים זה לזה – מיד לאחר ציון כתובת הזיכרון בה יושבת הלולאה, יופיעו הפעולות המחזוריות הכלולות בה, כמו גם קצב חזרתה.

אדגים את פעולת הלולאה המובנית הזו בעזרת רוטינת האסמבלי המשמיעה שוב ושוב את צלילי המוסיקה "עבדים היינו":

```
10     *=9600
20     PLA
30     LDA  #$0B
40     STA  $228
50     LDA  #$96
60     STA  $229
70     INC  $6FF
80     LDX  $6FF
90     LDA  $9700,X
100     CMP  #$FE
110     BCS  IPUS
120 CONT
130     STA  $D200
140     LDA  #$AA
150     STA  $D201
160     LDA  #$10
170     STA  $21A
180     RTS
190 IPUS
200     LDX  #$0
210     STX  $6FF
220     LDA  #$0
230     JMP  CONT
```

רוטינת האסמבלי לעיל בנויה כאמור משני מקטעים. להלן נכנה אותם "החלק המגדיר" ו"החלק המחזורי". אפתח דווקא בחלק המחזורי.

- **החלק המחזורי**

החלק המחזורי הוא בעצם נגן המוסיקה – מקטע קוד פשוט אשר קורא תו אחר תו ומנגן אותו. אם אינכם זוכרים כיצד מנוגנים צלילי המוסיקה בסביבת האסמבלי, שובו וקראו את הפרק האחרון של מחשבת 9.

שורה 70 מונה את תווי המוסיקה. לשם כך ייעדתי את תא הזיכרון האחרון בדף מספר 6, הלוא הוא: 1535 (או $6FF).

שורות 90-80 קוראות את תווי המוסיקה בזה אחר זה מתוך אזור בזיכרון בו הם שמורים – החל מכתובת 38656 (או $9700) ואילך.

שורות 110-100 תרות כל העת אחר הערך 255. ערך זה מציין כי מקטע המוסיקה הסתיים. כאשר המעבד פוגש את הערך 255,[4] הרוטינה קופצת לשורות 230-200, שם מתאפס המונה וקטע הנגינה מתחיל מההתחלה.

שורות 150-130 אחראיות לניגון הצליל המסוים.

- **החלק המגדיר**

שורות 60-30 מורות למעבד היכן ממוקם "החלק המחזורי". אם רוטינת האסמבלי יושבת החל מתא זיכרון 38400 (או: $9600), הרי שהחלק המחזורי ייכתב החל מכתובת הזיכרון הראשונה הפנויה – 38411 (או: $960B).

מכיוון שכך, הכתובת המצביעה על החלק המחזורי תהא: $960B. את הכתובת הזו עלינו למסור לזוג הבתים המצביעים: 552, 553 (או $228, $229). לתוך בית $229 יוכנס החלק המשמעותי בכתובת (MSD) – $96 ואילו אל תוך בית $228 יוכנס החלק הפחות משמעותי (LSD) – $0B.

4 שימו לב לשימוש שנעשה בפקודת הדילוג **BCS** (במקום: **BEQ**), וזאת משום שאנו משתמשים בצורת הפקודה: **LDA $9700,X**

בתים אלו – 552, 553 – אמונים על הלולאה המובנית, ומצטרף אליהם בית נוסף – הבית 538 (או $21A). בית זה מגדיר את **תדירות החזרה** של "החלק המחזורי". ככל שהערך שיוכנס לתוכו יהיה גדול יותר, כך **יקטן** קצב החזרה של החלק המחזורי. ערך 0 יפסיק את המחזוריות כליל ועל מנת לחדשה תידרש הפעלה מחדש של רוטינת האסמבלי.

בשורות 160-170, מוכנס אפוא הערך 16 אל תוך הבית 538. ערך זה יקטן במהלך המשחק ככל ששלבי המשחק יתקדמו, וכך קצב המוסיקה ילך ויגבר.

בטרם אחתום את הדיון ברוטינת נגינת המוסיקה, אתעכב על עניין נוסף.

מונה התווים (בשורה 70) יכול מטבעו לשאת כל ערך שבין 1 ל-255. מכיוון שכך, כמות הצלילים שניתן להשמיע בכל מחזור של נגינה מוגבלת ל-255 בלבד.

על פניו נראה שמדובר בכמות נרחבת של צלילים, ואולם יש לקחת בחשבון כי צלילי המוסיקה נכתבו ביחידות של רבעים (כלומר, כשמדובר בצלילים ארוכים מופע אותו ערך שוב ושוב, לעתים עד 4 פעמים ברצף!) ובנוסף יש גם יחידות של הפוגה מדי פעם בפעם.

עבור הקטע המוסיקלי "עבדים היינו", די היה בקצת פחות מ-255 יחידות צליל, ואולם עבור הקטע המוסיקלי "בצאת ישראל ממצרים" (מנגינת הניצחון בסיום המשחק) נדרשו למעלה מ-255 יחידות צליל.

על מנת להכפיל את סך הצלילים המנוגנים, בניתי רוטינה דומה, אך נדרשתי להכניס בה כמה שינויים.

להלן אציג את הרוטינה ואתעכב על השינויים בינה לבין קודמתה.

```
10    *=$636
20    PLA
30    LDA #$0
40    STA $CB
50    LDA #$96
60    STA $CC
70    LDA #$49
80    STA $228
90    LDA #$6
100    STA $229
110    INC $6FF
120    LDY $6FF
130    LDA ($CB),Y
140    CMP #$FE
150    BCS IPUS
160 CONT
170    STA $D200
180    LDA #$AA
190    STA $D201
200    LDA #$6
210    STA $21A
220    JMP END
230 IPUS
240    LDA $CC
250    EOR #$96
260    EOR #$97
270    STA $CC
280    LDX #$0
290    STX $6FF
300    LDA #$0
310    JMP CONT
320 END
330    RTS
```

שימו לב, את רוטינת נגן המוסיקה מיקמתי הפעם בדף 6, בכתובת הזיכרון 1590 ($636).[5] "החלק המחזורי" יושב החל מתא הזיכרון הפנוי הראשון – 1609 ($649), ולכן יש למסור ב"חלק המגדיר" את כתובת זו: אל תוך בית $229 נכנס החלק המשמעותי בכתובת (MSD) – $6 ואל תוך בית $228 נכנס החלק הפחות משמעותי (LSD) – $49.

את נתוני תווי המוסיקה שמרתי הפעם החל מכתובת 38400 (או $9600), אך במקום להפנות אל כתובת זו כשלעצמה, נעזרתי בשיטת "המיעון העקיף, Y" (ראו לעיל בסעיף 2 שבפרק א) – כתובת הזיכרון פורקה לשניים: את החלק המשמעותי (MSD) $96 (150) הכנסתי אל תוך הבית $CC (240), ואילו את החלק הפחות משמעותי (LSD) $0 הכנסתי אל תוך הבית $CB (239). (ראו שורות קוד 30-60).

שורות 230-310 מטפלות במקרה שבו פוגש המעבד בערך האיפוס (255). במקרה שכזה מתבצעות 2 פעולות: ראשית מתאפס כמובן המונה; ושנית מתעדכן הבית $CC (240) – אם מאוחסנת בו כתובת נמוכה ($96), היא מוגבהת לכתובת $97, ואם מאוחסנת בו כתובת גבוהה ($97), היא מונמכת חזרה לכתובת $96.

על מנת להחליף לסרוגין בין כתובת זיכרון אחת (החלק הראשון של קטע הנגינה) לבין כתובת זיכרון שנייה (החלק השני של קטע הנגינה) נעזרתי ברצף של שתי פקודות **EOR**. כאשר מבצעים שוב ושוב את הפעולה "או מוציא" עם הערך 150 ומיד אחר כך עם הערך 151, מתקבלים לסירוגין הערכים 150 ו-151.

5 מכיוון שמדובר בסיום המשחק "מכות מצרים", אין עוד צורך בתוכנות העזר שישבו בדף 6, וניתן לדרוס אותן ללא חשש.

פרק ב
בניית שלד המשחק

1. תזוזה אופקית של הפרעונים

במשחק "מכות מצרים" מתרוצצים לרוחב המסך 21 פרעונים (שלוש שורות של שבע דמויות) משמאל לימין, ואחר כך מימין לשמאל, ובכל אימת שסיימו מחזור אחד של תנועה (הלוך ושוב), הם יורדים שורה אחת למטה, פעם אחר פעם, ובסך הכול 14 פעמים (או יותר, כשמושמדת שורה שלמה של פרעונים).

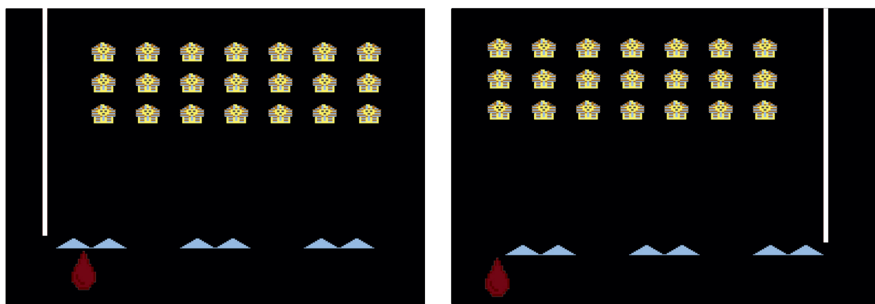
על מנת ליצור תנועה חלקה שאינה מושפעת מהתרחשויות אחרות (כגון תנועת דמות השחקן, ירי הטילים), בניתי את התנועה האופקית המחזורית של גוש הפרעונים כהפרעת VBI המתרחשת כל העת ברקע (כל עוד לא פוקדים עליה לעצור).

הפרעונים עצמם למעשה **אינם זזים כלל** לאורך כל המשחק, ומה שמשתנה היא כתובת הזיכרון המצביעה עליהם. כלומר, הסטה אחורה של כתובת הזיכרון ביחידה אחת, יוצרת **אפקט גלילה** של צעד אחד קדימה, והסטה קדימה של כתובת הזיכרון ביחידה אחת, יוצרת אפקט גלילה של צעד אחד אחורה.

כל עוד מספר הפרעונים (באחת השורות לפחות) הוא שבע, נדרשים שישה צעדים הלוך ושישה צעדים חזור על מנת להשלים מחזור אחד של תנועה, ואולם ככל שמצטמצם מספר הפרעונים (וזוהי הרי מטרת המשחק), מתארך המסלול שעובר גוש הפרעונים הנותר. כך למשל, כאשר נותר פרעון יחיד על המסך – נדרשים לו כבר 36 צעדים הלוך ו-36 צעדים חזור על מנת להשלים מחזור אחד של תנועה!

כדי לדעת מתי בדיוק לסובב את כיוונם של הפרעונים ולהתחיל "להתנועע" לצד הנגדי, רתמתי למשימה את אחד הטילים של ה-PM והפכתי אותו למעין קיר. את גובהו של ה"קיר" בניתי כגובה אזור המשחק (כלומר האזור בו יכולים להתנועע הפרעונים) ואת מיקומו האנכי הגדרתי כתלוי בכיוון תזוזת הפרעונים.

האיור הבא מדגים את מיקומו המשתנה של אותו "קיר":



(כשכיוון התנועה משמאל לימין) (כשכיוון התנועה מימין לשמאל)

את ה"קיר" צבעתי כמובן בצבע המסך, כלומר בשחור.

כל שנותר לי כעת לעשות הוא לבחון התנגשויות בין אותו קיר לבין הפרעונים, וכל אימת שמתרחשת התנגשות שכזו, לשנות את כיוון התנועה כמו גם את מיקומו של הקיר.

ועתה להצגת הקוד באסמבלי:

```
10      *=$600
20      PHP
30      PHA
40      TXA
50      PHA
60      TYA
70      PHA
```

```
80 ;   $6FA - NEXT_MOVE (INITIAL=20)
90 ;   $6FB - TEMPO
100 ; $6FC - VECTOR (200=R;50=L)
110 ; 6FD - LINE NUM.
120 ; 6FE - X-COR (100=START)
130 DLIST=$9055
140    LDA $14
150    SBC $6FA   ; EXIT UNLESS
160    CMP #$2    ; REACHING NEXT MOVE.
170    BCC DO
180    JMP FIN
190 DO
200    LDA $14
210    ADC $6FB     ; SETTING THE
220    STA $6FA     ; NEXT MOVE.
230    LDY $D000    ; CHECK COLLISION
240    CPY #$0      ; WITH "WALL".
250    BNE SWITCH   ; CHANGE DIRECTION.
260    LDY $6FC
270    CPY #$C8     ; IF WALL=200...
280    BEQ RIGHT
290    CPY #$32     ; IF WALL=50...
300    BEQ LEFT
310    JMP FIN      ; IF ERROR STATE...
320 LEFT
330    DEC $6FE     ; UPDATING X-COR.
340    LDX #$0
350    INX
360    INX
370 LULA
380    INX
390    INX
400    LDA DLIST,X
410    CLC
420    ADC #$1        ; INCREASING
430    STA DLIST,X    ; ALL SCREEN MEM.
440    INX            ; ADRESSES, LINE
450    LDA DLIST,X    ; BY LINE FROM 1ST
460    ADC #$0        ; TO 59TH SO THAT
470    STA DLIST,X    ; GRAPHICS WILL
480    CLC            ; MOVE BACKWORDS.
```

```
490    CPX #$3B
500    BCS FIN
510    JMP LULA
520 RIGHT
530    INC $6FE
540    LDX #$0
550    INX
560    INX
570 LULB
580    INX
590    INX
600    LDA DLIST,X
610    SEC
620    SBC #$1        ; DECREASING
630    STA DLIST,X    ; ALL SCREEN MEM.
640    INX            ; ADDRESSES, LINE
650    LDA DLIST,X    ; BY LINE FROM 1ST
660    SBC #$0ST,X    ; TO 59TH SO THAT
670    STA DLIST,X    ; GRAPHICS WILL
680    CLC            ; "MOVE" FORWORDS.
690    CPX #$3B
700    BCS FIN
710    JMP LULB
720 SWITCH
730    LDY #$1      ; CLEAR ALL
740    STY $D01E    ; COLLISIONS.
750    INC $6FD     ; HALF(!) LINE DOWN.
760    LDA $6FC
770    EOR #$C8     ; SWITCH BETWEEN 50
780    EOR #$32     ; AND 200.
790    STA $D004    ; UPDATE WALL PLACE.
800    STA $6FC
810 FIN
820    PLA
830    TAY
840    PLA
850    TAX
860    PLA
870    PLP
880    JMP $E462    ; BACK TO SYS. VBI
```

הסבר:

בשורות 70-20 נשמרים במחסנית ערכי הרגיסטרים, כמו גם תמונת מצב הדגלים, ובשורות 870-820 נשלפים ערכים אלו מן המחסנית ומוחזרים אל הרגיסטרים וגם מצב הדגלים שב לקדמותו.

שורות 180-140 דואגות "לזרוק" את המעבד החוצה מהפרעת ה-VBI כל עוד לא הגיע הרגע המדויק להניע את הפרעונים. אותו רגע מדויק מחושב כך: בתוך הבית $6FA (1786) שיועד למטרה זו, נשמרת בכל שלב שניית מחשב עתידית, המרוחקת מרחק קבוע מן השנייה הנוכחית. כאשר מגיע שעון המחשב אל אותו רגע ממש, פונה המעבד לבצע את ההפרעה ובד בבד מעודכן הבית $6FA ונשמרת בו שניית מחשב עתידית חדשה. אגב, השנייה העתידית ההתחלתית מחושבת ונשמרת בתוך הבית $6FA בתוכנת הבייסיק GAME.BAS, עוד לפני אתחול הפרעת ה-VBI (בשורה 590).

שורות האסמבלי 310-190 מבצעות שלוש משימות. ראשית, כאמור, מתעדכנת שניית המחשב העתידית הבאה. ערך המרחק הקבוע מן השנייה הנוכחית לשנייה הבאה שמור בתוך הבית הייעודי $6FB (1787) – וזהו למעשה קצב מהירות התנועה. לאורך המשחק משתנה ערכו של בית זה ככל ששלבי המשחק מתקדמים יותר.

שורות 250-230 בודקות אם אירעה התנגשות של הפרעונים עם אותו "קיר". אם התרחשה התנגשות, מנותבת הרוטינה אל מקום ייעודי שיטפל בכך.

השורות הבאות מנתבות את הרוטינה לאחד משני כיוונים – התקדמות שמאלה או התקדמות ימינה. אמנם, לא ייתכן מצב שבו לא תנותב הרוטינה לא שמאלה ולא ימינה, אך ליתר ביטחון הוספתי את שורה 180 שדואגות במקרה שכזה "לזרוק" את המעבד החוצה מהפרעת ה-VBI.

שורות 320-510 וכן 520-710 מטפלות בהנעת הפרעונים שמאלה או ימינה.

אסביר:
ראשית, מתעדכן הבית ‎$6FE‎ (1790). בבית הזה נשמר כל העת גודל ההסטה של גוש הפרעונים. הערך ההתחלתי נקבע על 100. ככל שגוש הפרעונים נע ימינה, עולה ערכו של הבית שוב ושוב ביחידה אחת, וככל שגוש הפרעונים נע שמאלה, יורד ערכו של הבית שוב ושוב ביחידה אחת. כזכור, במהלך המשחק יכולים הפרעונים לנוע עד 36 צעדים בכל כיוון, וכך ערכו המקסימלי של בית 1790 עשוי להיות 124, וערכו המינימלי עשוי להיות 88.

אחר כך מוזז כל גוש הפרעונים יחידה אחת – שמאלה, באמצעות פקודת החיבור **ADC**, או ימינה, באמצעות פקודת החיסור **SBC** (זכרו, כי כדי לגלול את הפרעונים ימינה, יש להסיט את מצביעי הזיכרון אחורה, וכדי לגלול את הפרעונים שמאלה, יש להסיט את מצביעי הזיכרון קדימה).

על מנת לדאוג שכל גוש הפרעונים יזוז כמקשה אחת, עוברת הרוטינה על כל נקודות העוגן – 21 במספר (או 23, כאשר מסולקת שכבת ההגנה של הפירמידות) ומקדמת אותן קדימה או אחורה ביחידה אחת. שימו לב, על מנת להזיז את מצביעי זיכרון המסך אני נעזר במצבו של דגל השארית C (פרק ג במחשבת 8 מסביר בהרחבה את הטכניקה הזו).

שימו לב גם לשורות 500-510 ו-690-700. כדי לבחון האם הרוטינה הגיעה לנקודת העוגן ה-21 (כלומר לפריט ה-60 ב-Display List) אני משתמש בפקודת ההתניה **BCS** (ולא **BEQ**), כלומר פוקד על המחשב לצאת מן הלולאה כאשר הוא מגיע לפריט **שאחרי** פריט 59 ב-Display List. הסיבה לכך היא השימוש בצורת הפקודה **LDA DLIST,X** שכזכור משפיעה על דגל השוויון Z (אם אינכם זוכרים, תוכלו לרענן את זיכרונכם בסוף סעיף ב בפרק "גלילה" שבמחשבת 9).

לבסוף, השורות 800-730 מטפלות ברגע החלפת כיוון התנועה. כזכור, כאשר מתנגש גוש הפרעונים עם הטיל המשמש כ"קיר", מתהפך כיוון התנועה ובד בבד, ממוקם הקיר במקומו החדש.

מכיוון שהקיר עשוי להימצא באחד משני מקומות – או בקורדינטה 200 או בקורדינטה 50, השתמשתי בשורות 780-770 ברצף של שתי פקודות EOR. כאשר מבצעים שוב ושוב את הפעולה "או מוציא" עם הערך 200 ומיד אחר כך עם הערך 50, מתקבלים לסירוגין הערכים 50 או 200.

לפני היציאה מן הרוטינה, מתעדכן בשורה 750 הערך השמור בבית הייעודי $6FD (1789). בבית הזה נשמר מספר ההחלפות של כיווני התנועה. מכיוון שאחרי כל שתי החלפות כיוון יורדים הפרעונים שורה אחת למטה, הרי שכדי לדעת מה מספר השורה העדכני, יש לחלק את הערך השמור בבית 1789 ב-2.

כשתנועת הפרעונים נעצרת

במהלך כל אחד משלבי המשחק הפרעונים מתנועעים כאמור הלוך ושוב לרוחב המסך. הם נעצרים מתנועתם רק במקרה שבו טיל האויב פגע בדמות השחקן, או כאשר גוש הפרעונים הגיע לאזור תנועת דמות השחקן.

מכיוון שתנועת הפרעונים מעוגנת, כפי שהסברתי בתוך הפרעת VBI הרי שהפסקת תנועת הפרעונים תושג מן הסתם על ידי ביטולה של אותה ההפרעה. דא עקה, הכנסת שינויים במנגנון ריענון המסך במהלך פעולת סריקת המסך של ה-Antic אינה דבר רצוי, והיא לעתים עשויה אף לגרום למערכת ההפעלה לקרוס.

על מנת למנוע מצב שכזה אימצתי את הטקטיקה של סנכרון הרוטינה שלי עם קצב ריענון המסך. כפי שלמדנו במחשבת 9 (בסעיף העוסק בגלילה עדינה),

ניתן לשלוח את המעבד לבצע משימה בדיוק ברגע סיום ריענון המסך, וזאת אם "נלכוד" את רגע תחלופת שניית המחשב. וכך, בדיוק באותו רגע ניתן להורות למחשב להפסיק או להתחיל מחדש את הפרעת ה-VBI, בלא חשש.

זוהי הרוטינה שכתבתי, שכל תפקידה הוא להדליק/לכבות את הפרעת ה-VBI:

```
10    *=$96C6
20    PLA
30    PLA
40    PLA
50    LDX $14
60    INX
70 LOOP
80    CPX $14
90    BNE DO
100    JMP LOOP
110 DO
120    CMP #$1
130    BEQ STARTER
140 KILLER
150    LDA #$55
160    STA $224
170    LDA #$C0
180    STA $225
190    JMP END
200 STARTER
210    LDA #$0
220    STA $224
230    LDA #$6
240    STA $225
250 END
260    RTS
```

הסבר:

שורות 40-20 שולפות מן המחסנית את ערכי הפרמטר שהתקבלו מן הבייסיק. הערך 1 ינתב את הרוטינה להדליק" את הפרעת ה-VBI , וכל ערך אחר ינתב את הרוטינה "לכבות" את ההפרעה.

שורות 100-50 לוכדות את רגע תחלופת שניית המחשב.

שורות 180-150 מכבות את הפרעת ה-VBI – הן מציבות מחדש את הערכים המקוריים שישבו בבתים 548, 549 (קרי: הערכים אשר יושבים בבתים אלו בסביבת בייסיק, מרגע אתחול המחשב).

שורות 240-210 מדליקות את הפרעת ה-VBI – הן מציבות את הערכים של רוטינת ההפרעה, ששמורה בדף 6.

שימו לב. הערכים המקוריים שישבו בבתים 548, 549 – אינם זהים בהכרח בכל מערכת הפעלה ועשויים להיות שונים בסביבת העבודה של אמולטור מסוים. מסיבה זו דאגתי **מיד עם הכנסת הרוטינה לדף 6** לעשות מניפולציה על קוד שפת המכונה, ולהכניס את הערכים הספציפיים של בתים אלו. עיינו בשורה 980 בקובץ: SEDER.BAS.

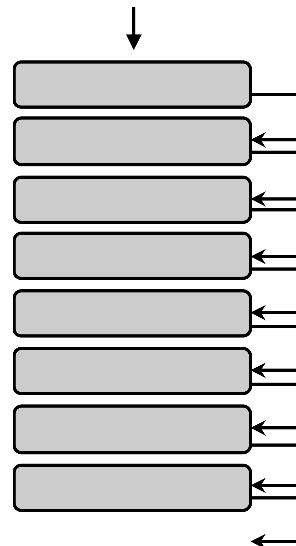
2. תזוזה אנכית של הפרעונים

הפרעונים כאמור אינם זזים כלל לאורך כל המשחק, וגם הירידה שלהם מטה מושגת כאפקט גלילה אנכי, על ידי מניפולציה על מצביעי זיכרון המסך.

גוש הפרעונים מתפרש למעשה על 8 יחידות גרפיות (כל אחת משלושת שורת הפרעונים בנויה משתי שורות, ועל אלו מתווספות 2 שורות רווח). לכן כל

שנדרש לעשות על מנת לייצר את אפקט הגלילה מטה הוא להעתיק שוב ושוב 8 נקודות עוגן קדימה – נקודת העוגן השמינית זזה למקום התשיעי; נקודת העוגן השביעית זזה למקום השמיני; וכן הלאה, עד לנקודת העוגן הראשונה שזזה למקום השני. לבסוף, יש לדאוג שנקודת העוגן הראשונה תצביע על אזור ריק בזיכרון.

האיור הבא ממחיש את פעולת ההעתקה של נקודות העוגן:



וזהו קוד האסמבלי:

```
10     *=$6A0
20 DLISTA=$8C58
30 DLISTB=$8C55
40     PLA
50     LDX #$38
60 LOOP
70     LDA DLISTB,X
80     STA DLISTA,X
```

```
90    DEX
100    LDA DLISTB,X
110    STA DLISTA,X
120    DEX
130    DEX
140    CPX #$3
150    BCC EXIT
160    JMP LOOP
170 EXIT
180    LDA #$98
190    STA DLISTA,X
200    DEX
210    LDA #$0
220    STA DLISTA,X
230    RTS
```

הסבר:

בשורות 30-20 מוגדרות 2 תוויות – אחת מכילה את כתובת ה-Display List, והשנייה, כתובת מתקדמת ב-3 מקומות. מכיוון שה-Display List בנוי (בחלקו הראשון והמרכזי) מנקודות עוגן בלבד, הרי שבין נקודת עוגן אחת למשנה יש מרווח של שלוש מקומות בדיוק.

כל שנותר לעשות אם כן, הוא להתקדם בעזרת שתי התוויות, ולהעתיק בזו אחר זו את כל נקודות העוגן. שורות 160-60 מבצעות פעולה זו בדיוק.

לבסוף, שורות 220-180 דואגות להציב בנקודת העוגן הראשונה אזור ריק בזיכרון, וזאת כדי להבטיח שלא יישארו חלקי פרעונים או כל גרפיקה אחרת בחלקו העליון של המסך.

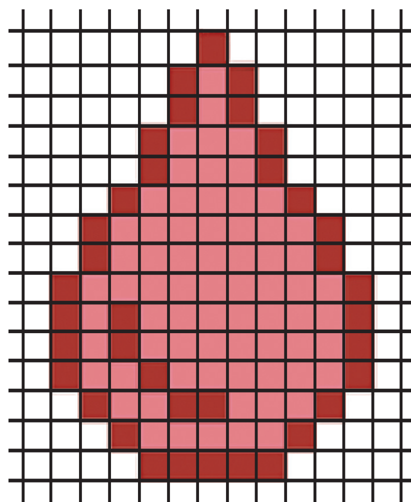
פרק ג
השחקן הנלחם בפרעונים

1. בניית מכות מצרים

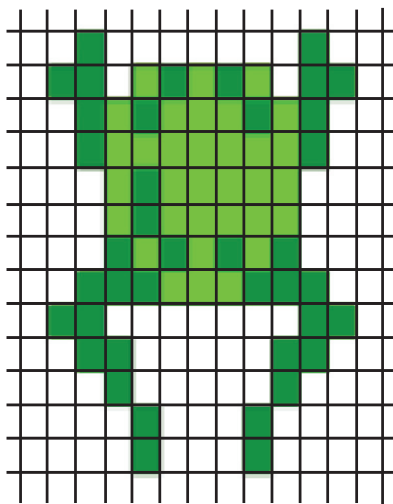
המשחק בנוי מעשרה שלבים, בדרגת קושי עולה – כנגד עשר מכות מצרים. בכל שלב גוברת מהירות תנועת הפרעונים ותדירות הטילים הנורים מהם.

דמות השחקן (כל אחת ממכות מצרים) בנויה מ-4 שחקנים של גרפיקת השחקנים (PM) – ניצבים לזה לצד זה או מולבשים זה על גבי זה. בכל שלב במשחק מתעדכנים נתוני הגרפיקה (לרבות הצבעים ורמת מתיחת הגרפיקה).

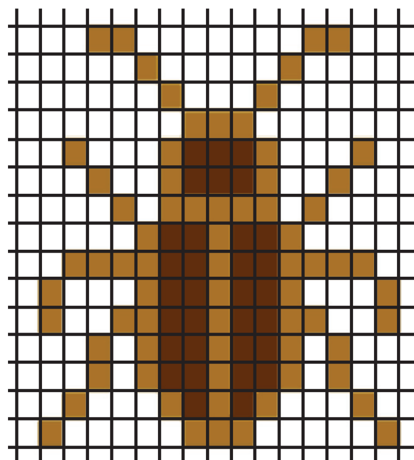
להלן שרטוט של כל אחד מעשרת השחקנים השונים:



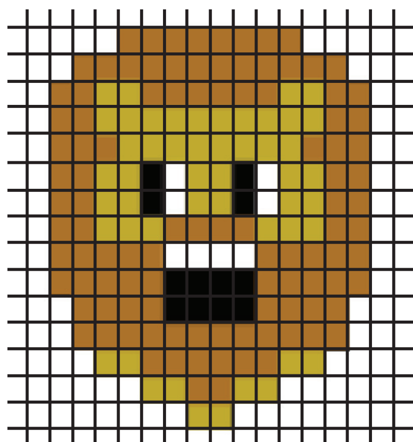
דם



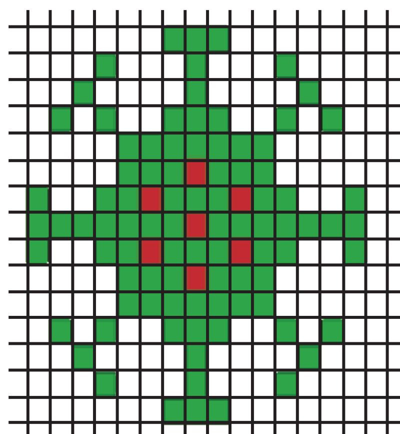
צפרדע



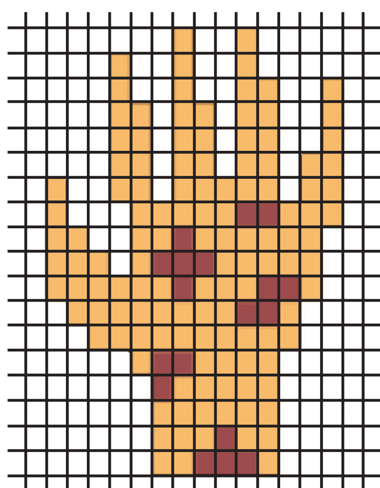
כינים



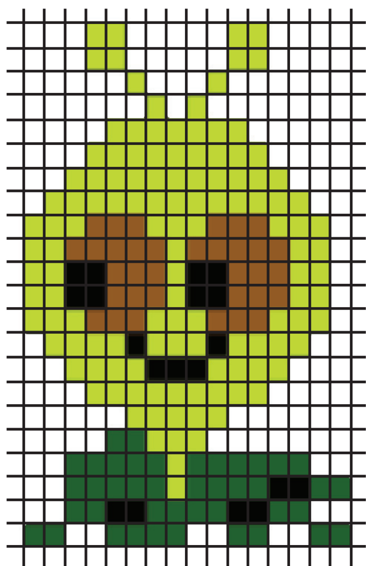
ערוב



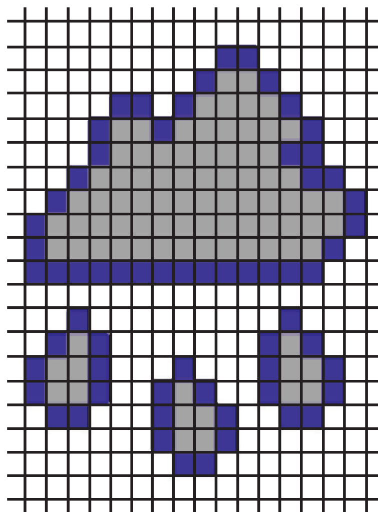
דבר



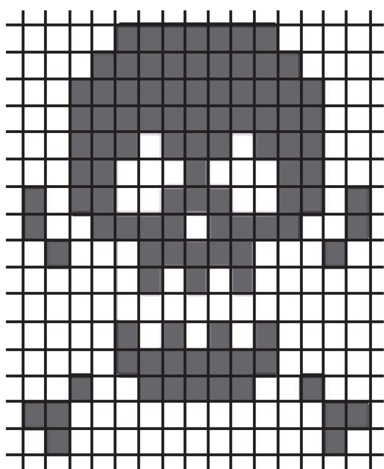
שחין



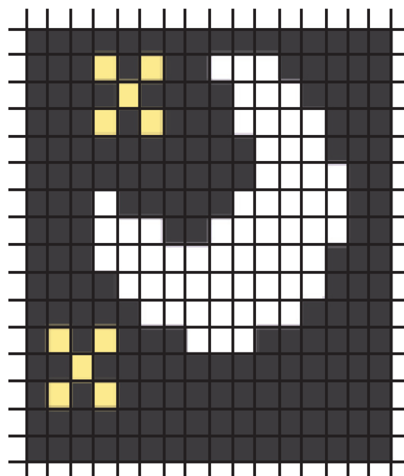
ארבה



ברד



מכת בכורות



חושך

על מנת להניע את דמות השחקן על הציר האופקי של המסך, נעזרתי ברוטינת אסמבלי קצרה. ניתן כמובן לבצע את המשימה בבייסיק בלבד, אולם מכיוון שמדובר בהזזה בו זמנית של ארבעה שחקני PM, רק רוטינת אסמבלי מבטיחה תזוזה חלקה, ולא ריקוד של חלקי הדמות.

זוהי אפוא הרוטינה שבניתי:

```
10    *=$6C5
20    PLA
30    PLA
40    PLA
50    STA  $D002
60    STA  $D000
65    ADC  #$0
70    STA  $D003
80    ADC  #$8
90    STA  $D001
100    RTS
```

שורות 40-20 מקבלות מן הבייסיק (באמצעות הפקודה USR) פרמטר אשר לפיו תמוקם הדמות על המסך.

שורות 90-60 פוקדות את הערכים הנדרשים אל תוך הבתים: 53248-53251 (הבתים האמונים על מקומם האופקי של השחקנים).

שימו לב. ישנן דמויות (כגון "דם"), אשר שניים מן השחקנים המרכיבים אותן מוסטים ימינה, וישנן דמויות (כגון "חושך") אשר רק אחד מהשחקנים המרכיבים אותן מוסט ימינה. לצורך כך, הכנסתי – לפי הצורך – מניפולציה מתוך הבייסיק על רוטינת שפת המכונה – מניפולציה אשר מצריחה למעשה בין שורות 65 ו-80.

2. יורים טילים

מטרת המשחק היא להשמיד את כל הפרעונים. לצורך כך, יכולה דמות השחקן הפעילה (אחת ממכות מצרים) לשגר לעברם טילים. בכל רגע נתון יכול להימצא "באוויר" טיל אחד בלבד, ולצורך כך מיועד מעין דגל – משתנה בשם TIL, אשר נדלק (מקבל את הערך 1) כאשר הטיל "באוויר" ונכבה (מקבל את הערך 0) כאשר הטיל מסיים את מעופו.

הטיל הידידותי (הנורה לעבר הפרעונים) מתקדם מלמטה למעלה. כזכור ממחשבת 7 בפרק העוסק בטילים, על מנת להניע את הטיל על הציר האנכי יש להעתיק ממקומם את כל הפיקסלים (סיביות) הבונים אותו ובמקביל, לדאוג לאפס את המיקום הקודם (המנוגד לכיוון תנועת הטיל) בו שהה הטיל.

אלא, שבמשחק שלנו משתתף יותר מטיל אחד. בנוסף לטיל הידידותי שיורה הדמות, נורים גם טילי אויב מן הפרעון כלפי מטה, ויש גם את הטיל שדנו בו בסעיף הקודם (אותו "קיר" שחור התוחם את תנועת הפרעונים).

על מנת לוודא שטיל אחד לא יפריע למעופו של טיל אחר, חייבים להימנע מהעתקה פשוטה של פיקסלים כמו גם מאיפוס פשוט של פיקסלים, אלא להתחשב בפיקסלים של טילים אחרים המתנועעים במקביל על המסך.

? כיצד עושים זאת?

מנצלים את הידע שרכשנו בפרק "לוגיקה מדליקה (וגם מכבה)" במחשבת 8.

כזכור, ישנן שתי פקודות לוגיות באסמבלי המאפשרות הדלקה וכיבוי של סיביות: הפקודות ORA ו-AND. על מנת להניע את הטילים השונים (הן טיל הפגיעה והן טיל האויב) – בניתי רוטינת אסמבלי המקבלת מן הביסייק מידע על אודות זהות הטיל, ומיקומו על המסך.

:זוהי הרוטינה

```
10     *=$9689
20  TOY=$9990
30  TIL=$9980
40     PLA
50     PLA
60     PLA
70     TAY
80     PLA
90     PLA
100     TAX
110     CPY #$30
120     BEQ OYEV
130     CPY #$C
140     BEQ MYTIL
150     JMP END
160  OYEV
170     LDA TOY,X
180     AND #$CF
190     STA TOY,X
200     INX
210     LDA TOY,X
220     AND #$CF
230     STA TOY,X
240     INX
250     LDA TOY,X
260     AND #$CF
270     STA TOY,X
280     INX
290     LDA TOY,X
300     ORA #$30
310     STA TOY,X
320     JMP END
330  MYTIL
340     LDA TIL,X
350     ORA #$C
360     STA TIL,X
370     INX
380     LDA TIL,X
```

```
390    AND #$F3
400    STA TIL,X
410    INX
420    LDA TIL,X
430    AND #$F3
440    STA TIL,X
450    INX
460    LDA TIL,X
470    AND #$F3
480    STA TIL,X
490    JMP END
500 END
510    RTS
```

הסבר:

בשורות 100-40 נשלפים מן המחסנית שני הפרמטרים אשר נשלחו אל הרוטינה מן הבייסיק. בריגיסטר Y מאופסנת זהות הטיל שבו תטפל הרוטינה. הערך 48 (או 00110000 בשיטת הספירה הבינארית) הוא טיל מספר 2 – טיל האויב; ואילו הערך 12 (או 00001100 בשיטת הספירה הבינארית) הוא טיל מספר 1 – הטיל משמיד הפרעונים. בריגיסטר X מאופסן מיקומו של הטיל הנדון על הציר האנכי.

שורות 150-110 מנתבות את הרוטינה לטפל באחד משני הטילים. כל ערך אחר זולת 12 ו-48 "יזרוק" החוצה מן הרוטינה.

שורות 320-170, ובהמשך 490-340 מטפלות בהזזת הטיל על הציר האנכי. אסביר להלן על הטיל הידידותי (הנורה לעבר הפרעונים). הטיפול בטיל האויב דומה מאוד.

מכיוון שהטיל הידידותי מתקדם לעבר הפרעונים מלמטה למעלה, יש להדליק בשלב ראשון את הפיקסלים במיקום שנמסר לרוטינה. לשם כך,

נעזרים בפקודה ORA ומבצעים את הפעולה עם הערך 00001100. אחר כך יורדים מיקום אחד, ומכבים את הפיקסלים שהטיל הותיר קודם. לשם כך, נעזרים בפקודה AND ומבצעים את הפעולה עם המשלים הלוגי לערך 00001100, הלוא הוא 11110011, כלומר 243. יורדים עוד מיקום אחד וחוזרים על הפעולה בשנית, ושוב – בשלישית (וזאת משום שהטיל המהיר קופץ בקפיצות של 3).

3. ... ופוגעים בפרעונים

מכיוון שהפרעונים המתנועעים על המסך הם אלמנטים גרפיים, הדרך האפקטיבית לבחון התנגשויות בינם לבין הטילים הנורים אליהם היא לנטר כל העת (כל עוד הטיל הנורה נמצא "באוויר") את הבית 53249, הבוחן התנגשויות בין טיל לאלמנט גרפי.

כאשר מאותרת פגיעה שכזו, כלומר כאשר ערכו של הבית שונה מ-0, צריכים לקרות שלושה דברים: (א) הטיל הפוגע צריך לסיים את תנועתו ולהעלם מן המסך; (ב) הפרעון שנפגע, גם הוא צריך להיעלם מן המסך (אגב צליל קצר); (ג) ממצבת הפרעונים צריך להיגרע פרעון אחד.

משימות א ו-ג קלות לביצוע. משימה ב לעומתן טריקית למדיי. כזכור, הפרעונים כשלעצמם אינם זזים כלל, ומה שמתנועע הם הבתים המצביעים על זיכרון המסך. דא עקה, על מנת לקבוע איזה פרעון בדיוק נפגע, נדרש חישוב די מורכב: עלינו להפחית מן הקואורדינטות של נקודת הפגיעה את ההיסט (הן ההיסט האופקי והן ההיסט האנכי) של הבתים המצביעים, אך כאן יש להביא בחשבון כי לדמות הפרעון יש רוחב משלה והפגיעה יכולה להיות לא רק במרכז הפרעון אלא לכל רוחבו. זאת ועוד זאת, מכיוון שתנועתו של

הפרעון מתרחשת כאמור ברקע (כהפרעת VBI), אי אפשר לדעת היכן בדיוק יימצא הפרעון ברגע בו **מחושב** מקום הפגיעה.

על מנת לפתור בעיה זו, בניתי בבייסיק כמה שורות קוד המחשבות את **הטווח** בו אמור להימצא כל פרעון ברגע הפגיעה בו (שורות 1210-1090 בקובץ GAME.BAS). שימו לב לשימוש שעשיתי בפקודה **ABS** (המשמשת למציאת ערך מוחלט). זאת מכיוון שהטווח שבו עשוי הטיל לפגוע במטרה משתרע החל מקצהו הימני ועד לקצהו השמאלי של הפרעון.

בכל אופן, חישוב טווח הפגיעה האפשרית מצליח להניב תוצאת אמת ברוב רובם של המקרים, אך יש מקרים (ספורים) בהם החישוב יצביע על פרעון אחר, אשר לא באמת נפגע.

על מנת למנוע מצב בו ייגרע פרעון ממצבת הפרעונים מבלי שאכן סולק כזה מן המסך – הוספתי כמה שורות בבייסיק (שורות 1277-1271) שמוודאות כי במקום שבו מסולק פרעון מן המסך אכן הופיע קודם פרעון. אם לא – מעודכנת בחזרה מצבת הפרעונים והמשחק נמשך כאילו לא התרחשה פגיעה.

באשר למשתמש הקצה המשחק במשחק – הוא עשוי להיתקל באחד משני מצבים לא רצויים: מצב שבו הטיל אשר ירה פגע בפרעון אך השמיד פרעון **אחר** סמוך לו; או מצב שבו הטיל אשר ירה פגע בפרעון, אך לא הוביל להשמדתו (מעין פרעון עם חסינות יתר). כאמור, מדובר באירועים נדירים למדיי, והם עשויים להתרחש רק כאשר פגיעת הטיל הידידותי היא באחד הקצוות של הפרעון.

פרק ד
הפרעונים משיבים מלחמה

לפרעונים יש שתי דרכים להביס את דמות השחקן – האחת היא להתקדם ולהגיע עד לשורה התחתונה; והשנייה היא לפגוע בו באמצעות טיל. להלן אסביר כיצד נורים טילים מן הפרעונים אל עבר דמות השחקן.

1. טילי אויב אפקטיביים

ההחלטה **מתי** לשגר טיל, מותנית ראשית כול, בכך שאין טיל אויב אחר שנמצא באותו הזמן "באוויר". לשם כך ייעדתי "דגל" תלת-מצבי – משתנה בשם: POY1 אשר עשוי לקבל את הערך 2 כאשר טיל האויב "באוויר"; את הערך 0 כאשר טיל האויב מסיים את מעופו; או את הערך 1 (להלן אסביר מתי ולשם מה).

כאשר ערכו של המשתנה POY1 הוא 0, נערכת הגרלה (באמצעות פקודת הבייסיק **RND**) וברגע רנדומלי מוחלט על שיגור טיל אויב.

ההחלטה **מהיכן** לשגר טיל, נקבעת לפי מיקום הימצאה באותה עת של דמות השחקן. כך הופך טיל האויב להיות אפקטיבי, כלומר בעל פוטנציאל גבוה לפגיעה.

? כיצד מתכנתים זאת?

ראשית, משעה שהוחלט (באמצעות אותה הגרלה) לשגר טיל אויב, מקבל המשתנה POY1 את הערך 1, והקואורדינטה האנכית של טיל האויב מתעדכנת לפי מיקומה של דמות השחקן ונשמרת בתוך המשתנה XOY1.
(ראו שורות 780-760 בקובץ: GAME.BAS)

טיל האויב ברגע זה הוא בעצם קיר ארוך, כגובהו של אזור המשחק וצבעו שחור, כצבע המסך. מטרת אותו "קיר" היא לאתר פרעון על הציר האנכי. כאשר מאותר פרעון (בשורת הבייסיק 790), מתעדכן המשתנה POY1 ומקבל את הערך 2, וזהו הסימן למחוק את אותו קיר ארוך, ולהופכו לטיל שמתחיל להתנועע מטה, לעבר דמות השחקן.

אותו "קיר מאתר", אגב, מבטיח לא רק שטיל האויב יהיה אפקטיבי וקטלני. הוא גם מבטיח בראש ובראשונה שאותו הטיל לא ישוגר ממקום סתמי על המסך, אלא רק ממקום שבו ניצב באותו רגע פרעון.

על מנת לעבור במהירות ממצב של "קיר מאתר" לטיל נורה ולהיפך, בניתי רוטינה נוספת באסמבלי, המקבלת מתוכנת הבייסיק את הקוד 1, המורה למתוח את הקיר המאתר, או את הקוד 2, המורה לנקות את הקיר המאתר:

```
10    *=$9630
20 TIL=$9980
30    LDX #$5E
40    PLA
50    PLA
60    PLA
70    CMP #$1
80    BEQ SECON
90    CMP #$2
100    BEQ SECOFF
110    JMP END
120 SECON
```

```
130     LDA TIL,X
140     ORA #$30
150     STA TIL,X
160     DEX
170     CPX #$3
180     BCC END
190     JMP SECON
200 SECOFF
210     LDA TIL,X
220     AND #$CF
230     STA TIL,X
240     DEX
250     CPX #$3
260     BCC END
270     JMP SECOFF
280 END
290      RTS
```

הסבר:

על מנת למתוח את הקיר, השתמשתי (בשורה 140) בפקודה הלוגית **ORA** המדליקה סיביות, כלומר את הפיקסלים של הטיל. על מנת למחוק את הקיר, השתמשתי (בשורה 220) בפקודה הלוגית **AND** המכבה סיביות, כלומר את הפיקסלים של הטיל.

2. שכבת הגנה של פירמידות

בתחתית המסך נמצאת כאמור שורת פירמידות (המצוירת בשורות 470-570 בקובץ: GAME.BAS), והיא מהווה שכבת הגנה. כאשר טיל האויב פוגע בפירמידה, הוא שוחק אותה במקום לפגוע בדמות השחקן. פגיעות נוספות של טילי אויב באותה הנקודה יחשפו את דמות השחקן לפגיעה.

הפירמידות מצוירות ביחידות גרפיקה מסוג 14, ועל מנת לשחוק אותן השתמשתי בפקודות הבייסיק: PLOT ו-DROWTO. זאת אחרי שהורתי ל-Antic באיזו גרפיקה מדובר, באמצעות הפקודה: POKE 87,14 (אם אינכם זוכרים מדוע, רעננו את זיכרונכם בפרק הראשון של מחשבת 9).

עתה נדרשת הכרעה: כמה פיקסלים לשחוק מן הפירמידה בנקודת הפגיעה. ייתכנו שני מצבים: אם פגיעת הטיל היא בנקודה גבוהה של הפירמידה, היא תיקטם אך לא תחשוף את דמות השחקן; אך אם פגיעת הטיל היא בנקודה נמוכה של הפירמידה, ייפער בה חור שיחשוף את דמות השחקן.

לצורך איתור נקודת הפגיעה וחישוב מספר הפיקסלים שיש לגרוע מן הפירמידה, השתמשתי בפקודה הבייסיק LOCATE (ראו שורות 860-870 בקובץ: GAME.BAS).

ראשית, אסביר את מהות הפקודה LOCATE. פקודה זו נכתבת בבייסיק כך:

```
LOCATE X,Y,A
```

X ו-Y הן קואורדינטות על פני המסך הגרפי הפעיל (בדיוק אותן קואורדינטות המשמשות בפקודות הבייסיק: PLOT ו-DROWTO).

במשתנה A נשמר המידע שמכילה אותה נקודה על גבי המסך – 0 אם הפיקסל כבוי; 1 אם הפיקסל בצבע הראשון; 2 אם הפיקסל בצבע השני וכו'.

הפקודה LOCATE יעילה במיוחד במשחקי פעולה בהם יש אויב הרודף אחרי השחקן. באמצעות פקודה זו, יכול האויב לנטר בכל רגע נתון את סביבתו ולפעול לפי הנתונים המתקבלים.

על מנת לחשב את מספר הפיקסלים שיש לגרוע מן הפירמידה, מניתי את מספר הפיקסלים הדולקים מתחת לטיל האויב באזור מיקומם של הפירמידות. הבחנתי בין שלושה מצבים: (א) אין אף פיקסל מתחת לטיל האויב – משמע, אין פירמידה לשחוק; (ב) יש פחות מ-5 פיקסלים של פירמידה מתחת לטיל האויב; (ג) יש 5 פיקסלים או יותר מתחת לטיל האויב.

שורות הבייסיק 950-980, מטפלות בשני המצבים האחרונים וגורעות את כמות הפיקסלים הנדרשת מן הפירמידה הנפגעת.

3. כאשר שכבת ההגנה מסולקת...

המשחק "מכות מצרים" בנוי כך שכאשר גוש הפרעונים מתקרב מאוד אל דמות השחקן, שכבת ההגנה של הפירמידות מסולקת מהמסך וניתנת לדמות השחקן הזדמנות אחרונה לפגוע בפרעונים הנותרים, לפני שיספיקו להתקדם עוד 2 שורות לעברו.

סילוק שורת הפירמידות כרוכה בשלוש מניפולציות: (א) מניפולציה על ה-Display List ; (ב) מניפולציה על אופי התנהגותם של הטילים השונים; (ג) מניפולציה על רוטינות האסמבלי האמונות על תנועת הפרעונים. אפרט:

(א) מניפולציה על ה-Display List

כזכור, ה-Display List בתחילת המשחק מחלק את המסך לשני אזורים – אזור הפרעונים – יחידות גרפיות מסוג 12, ואזור הפירמידות – יחידות גרפיות מסוג 14. סילוק הפירמידות דורש סילוק של 12 יחידות מסוג 14, והצבת 4 יחידות מסוג 12, במקומן. כמובן שיש לעדכן גם את הבתים המצביעים על

יחידות אלו, ולבסוף, יש לעדכן את מיקומה של פקודת ה-**DLI** (קרי: הסיבית השמינית). על משימות אלו אחראיות שורות הבייסיק 1300-1320.

(ב) מניפולציה על אופי התנהגותם של הטילים השונים

במהלך המשחק ייתכן אחד משני מצבים – שורת פירמידות קיימת או לא קיימת. לצורך אבחנה בין שני מצבים אלו, ייעדתי משתנה בשם CHK, המשמש מעין דגל. ערך 1 מציין שיש שורת פירמידות, וערך 0 מציין ששורת הפירמידות סולקה.

נפתח בתנועתו של **טיל האויב**. שורת הבייסיק 860 מאפיינת את התנהגותו של טיל האויב באזור הפירמדיות. אם קיימת שורת פירמידות (קרי ערכו של CHK הוא 1), מחושבת כמות השחיקה שלהן. מאידך, אם סולקה שורת הפירמידות, הטיל ממשיך להתקדם מטה אל עבר דמות השחקן.

לגבי **הטיל ה"ידידותי"**: מכיוון שהפרעונים והפירמידות הם אובייקטים גרפיים מאותו סוג (בניגוד לגרפיקת השחקנים), כאשר הטיל הידידותי מתנגש באובייקט גרפי חייבים להבחין בין התנגשות אפקטיבית של הטיל בפרעון, לבין התנגשות של הטיל בפירמידה (שכמובן לא אמורה להשפיע עליה). גם כאן בא לעזרה המשתנה CHK בשורות הבייסיק 1030-1040.

(ג) מניפולציה על רוטינות האסמבלי

בתנועת הגלילה של הפרעונים על המסך מטפלות כאמור שתי רוטינות אסמבלי – רוטינת הפרעת VBI האחראית על הגלילה האופקית; ורוטינה נוספת האחראית על הגלילה האנכית.

בשתי הרוטינות תנועה הגלילה מוגבלת לאזור הפעיל של הפרעונים בלבד, שהרי לא נרצה שגם הפירמידות יתנועעו מצד אל צד, או שיידחקו אט אט כלפי מטה!

מסיבה זו הוספתי את שורת הבייסיק 1330 אשר מעדכנת **בתוך רוטינות שפת המכונה** את טווח הגלילה הרצוי. למותר לציין כי הכנסת מניפולציה על רוטינת שפת מכונה במהלך פעולתה השוטפת דורשת זהירות ותשומת לב, אחרת מערכת ההפעלה עלולה לקרוס ולחייב הפעלה של המחשב מחדש.

4. מתי שכבת ההגנה מסולקת?

על פניו נראה שהתשובה לכך פשוטה. הפירמידות מסולקות כאשר הפרעונים נמצאים במרחק שתי שורות מדמות השחקן. ואולם, יש לזכור כי גוש הפרעונים בנוי מ-3 שורות. אם שורת הפרעונים התחתונה (השלישית) הושמדה, הרי שלגוש הפרעונים הנותר יש שתי שורות נוספות לרדת עד שיגיעו לדמות השחקן. ואם גם שורת הפרעונים האמצעית (השנייה) הושמדה, הרי ששוב לגוש הנותר יש שתי שורות נוספות לרדת עד שיגיע אל דמות השחקן.

שלושה משתנים ייעודיים: ROW1, ROW2, ROW3 וכן שני המשתנים CHKLINE ו-GOVL נועדו לטפל בדיוק בעניין זה.

עם הצבת 21 הפרעונים בתחילת כל שלב, מאותחלים שלושת המשתנים ROW1, ROW2, ROW3 ומקבלים את הערך 7. כל אימת שמושמד פרעון מן השורה התחתונה, פוחת הערך השמור במשתנה ROW3, כך גם באשר לשורות השנייה ולשורה הראשונה.

המשתנה CHKLINE מכיל את ציון השורה על המסך בה יסולקו הפירמידות, ואילו המשתנה GOVL מכיל את ציון השורה על המסך שבה יפגוש גוש הפרעונים את דמות השחקן ויגרום לפסילתו.

כאשר השורה השלישית אינה כוללת עוד פרעונים (כלומר: ROW3=0), מתעדכנים שני המשתנים CHKLINE ו-GOVL; וכאשר גם השורה השנייה אינה כוללת עוד פרעונים, מתעדכנים שוב שני המשתנים הללו.

ראו שורות 660-670 בקובץ GAME.BAS.

פרק ה
תמונת ניצחון

עם סיום כל שלבי המשחק – עשרה במספר – מגיעה סוף כל סוף "תמונת הניצחון" – על המסך מופיעה פירמידה קטנה ושש דמויות צבעוניות צועדות כשגביהן אליה – כסמל ליציאת מצרים. בחלק העליון של המסך מתנוסס הכיתוב: "בצאת ישראל ממצרים" עם אפקט אנימציה של צבעי הקשת, וברקע מנוגנת המוסיקה "בצאת ישראל ממצרים".

מכיוון שמדובר בסיום המשחק, אין עוד צורך בתוכנות העזר שישבו בדף 6 (כגון: הפרעת ה-VBI; רוטינת הגלילה האנכית של הפרעונים וכו'). לכן דרסתי אותן ומיקמתי בדף 6 תוכנות עזר חדשות, רלוונטיות למסך הסיום.

שורות הקוד האמונות על אפקט האנימציה של קשת הצבעים הוסברו היטב במחשבת 9 והקוראים מוזמנים לעיין שם, בפרק העוסק בהפרעות DLI.

על שורות הקוד האמונות על מוסיקת הרקע (השיר "בצאת ישראל ממצרים") כבר הסברתי לעיל בפרק א'. את תווי המוסיקה טענתי הפעם בשיטה המהירה: את כל ערכי הצלילים המרתי לרצף של תווי ASCII ושמרתי אותם במשתנה מחרוזתי ארוך. אחר כך העתקתי את אזור הזיכרון הדינאמי של המשתנה המחרוזתי לאזור זיכרון קבוע אשר ממנו תנוגן המוסיקה. עשיתי זאת באמצעות רוטינת אסמבלי:

```
10    *=$600
20 LMEM=$9600
30 HMEM=$9700
40     LDY #$0
```

```
50    PLA
60    PLA
70    STA $CC
80    PLA
90    STA $CB
100    PLA
110    PLA
120    CMP #$1
130    BEQ LOW
140    CMP #$2
150    BEQ HIGH
160    JMP EXIT
170 LOW
180    LDA ($CB),Y
190    STA LMEM,Y
200    CMP #$FE
210    BCS EXIT
220    INY
230    JMP LOW
240 HIGH
250    INC $CC
260 LOOP
270    LDA ($CB),Y
280    STA HMEM,Y
290    CMP #$FE
300    BCS EXIT
310    INY
320    JMP LOOP
330 EXIT
340    RTS
```

הסבר:

שורות 90-40 שולפות מן המחסנית 2 פרמטרים (המתקבלים מתוכנת הבייסיק WINNER.BAS). הפרמטר הראשון הוא הכתובת הדינאמית של המשתנה המחרוזתי. הפרמטר השני הוא הערך 1 או 2.

הערך 1 מנתב את הרוטינה אל שורות 230-170, אשר אחראיות לקריאת החלק הראשון של נתוני המוסיקה (254 הצלילים הראשונים). הערך 2 מנתב את הרוטינה אל שורות 320-240, אשר אחראיות לקריאת החלק האחרון של נתוני המוסיקה.

שימו לב: כתובת הזיכרון הדינאמית (שהתקבלה כאמור כפרמטר) מוכנסת אל זוג הבתים המצביעים 203, 204. אחר כך, באמצעות שיטת "מיעון עקיף, Y" פונה המעבד אל הכתובת הזו ומעתיק את 254 הבתים העוקבים למקומם הקבוע בזיכרון. בקריאה נוספת לרוטינה – הפעם עם פרמטר 2 – כל שיש לעשות הוא להגביה את הבית המצביע 204 ביחידה אחת. כך, מופנה הפעם המעבד לכתובת המרוחקת מקודמתה ב-256 צעדים, ומעתיק את הבתים העוקבים הנותרים למקומם הקבוע בזיכרון.

שרטוט הפירמידה הקטנה

את הפירמידה הקטנה המופיעה בצד השמאלי של המסך בניתי באמצעות צירוף של ארבעה שחקנים (PM), כולם צמודים זה לזה ובאותו הצבע.

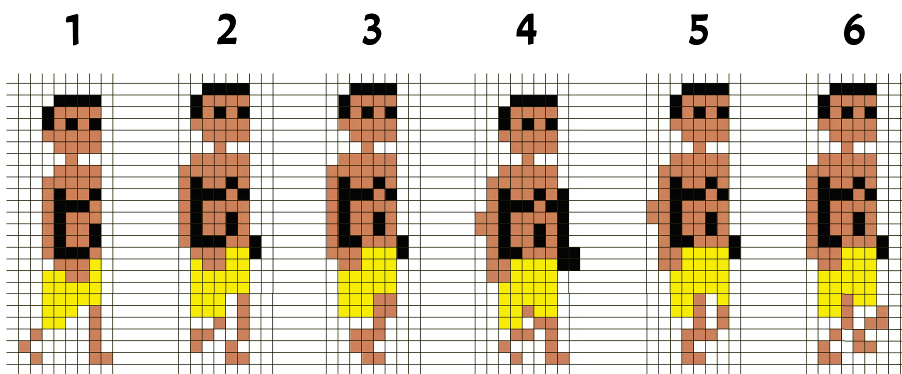
לשם כך, נדרשתי קודם כל לנקות את כל הנתונים הקודמים שנשמרו באזור הזיכרון של גרפיקת השחקנים. כדי לעשות זאת במהירות, בניתי רוטינת אסמבלי קצרה:

```
10     *=$686
20     PLA
30     LDA #$0
40     LDX #$0
50 LOOP
60     STA $9800,X
70     STA $9900,X
80     STA $9A00,X
90     STA $9B00,X
100     INX
110     CPX #$FE
120     BCS END
130     JMP LOOP
140 END
150     RTS
```

שורות 130-50 עובדות על ארבעת הדפים המרכיבים את אזור הזיכרון של גרפיקת השחקנים ומאפסות את תוכנם.

הדמויות הצועדות

כדי ליצור אנימציה של איש הולך שרטטתי שש פוזיציות של תנועה. הגבלתי את עצמי לשימוש בשלושה צבעים בלבד, וזאת על מנת לייצר את הדמות על יחידות גרפיקה מסוג 13. זהו השרטוט המונח על רשת קווים:



כל אחד משש מצבי הצעידה השונים, בנויה משש מטריצות – כל אחת בגודל של 4X8 משבצות, שהם למעשה מטריצות של 8X8 סיביות (זכרו כי על מנת ליצור תווים צבעוניים משתמשים בשיטה ה"בזבזנית", בה כל שתי סיביות מייצרות פיקסל אופקי אחד!)

על פניו נראה כי נדרשים אפוא 36 תווים כדי לאחסן את רצף התנועה כולו, אולם נקל לראות כי יש אזורים בדמות המתנועעת אשר חוזרים על עצמם. כך למשל אזור הראש זהה בארבעת המצבים (6,5,3,2), וחוזר על עצמו בשינוי קל גם בשני המצבים הנותרים (4,1).

בסך הכול נזקקתי אפוא, ל-26 תווים על מנת להכיל את כל מצבי הדמות. נתוני עיצוב התווים מופיעים בשורות 850-1100 שבקובץ: WINNER.BAS.

המסך הגרפי שבו מוצגת "תמונת הניצחון" הוא מסוג 2, אך במרכזו מתחתי 3 יחידות גרפיות מסוג 13 – וזאת על מנת להציב שם את הדמויות הצועדות.

את מצבי הצעידה השונים – ששה במספר, מיקמתי בתוך תא מחרוזתי בן 720 בתים, לפי החישוב הבא: אם כל יחידה גרפית מסוג 13 מתפרשת על 40

תאי זיכרון, הרי ששלוש יחידות כאלו מתפרשות יחד על 120 תאים; ואם נכפיל זאת בששה מצבי אנימציה, נקבל בסך הכול 720 תאים.

כל שנותר לעשות הוא לעדכן אחת לזמן מה את זוג מצביעי זיכרון המסך של "נקודת העוגן" הפותחת את יחידת הגרפיקה מסוג 13 ולכוון אותם אל המקטע הרלוונטי בזיכרון הדינאמי של התא המחרוזתי.

כך למשל, במצב האנימציה הראשון, יש לכוון את זוג מצביעי זיכרון המסך לנקודת ההתחלה של המערך המחרוזתי. במצב האנימציה השני יזוזו מצביעי זיכרון המסך 120 יחידות קדימה; במצב האנימציה השלישי, ינועו מצביעי זיכרון המסך למקום ה-240 במערך המחרוזתי וכן הלאה...

כדי לקבל תנועה מהירה וחלקה במיוחד, בניתי רוטינת אסמבלי קצרה שכל תפקידה הוא לעדכן את זוג הבתים המצביעים:

```
10      *=$650
20  ADLIST=$9365
30  BDLIST=$9366
40      PLA
50      PLA
60      TAX
70      PLA
80      STA  ADLIST
90      STX  BDLIST
100      RTS
```

הרוטינה מקבל מתוכנת הבייסיק פרמטר – את כתובתו הארעית של המשתנה המחרוזתי, בתוספת ההסטה קדימה (לפי מצב האנימציה הרלוונטי) ומעדכנת לפיו את זוג הבתים המצביעים ב-Display List.

שימו לב ליתרון נוסף לשימוש ברוטינת האסמבלי הנ״ל: שליחת הפרמטר מן הבייסיק כשלעצמה פירקה את הכתובת לזוג בתים מצביעים, אשר אותם ניתן להציב כסדרם כמצביעי זיכרון המסך. אילו ביצענו את המניפולציה בבייסיק, נדרש היה להמיר את הכתובת השלמה לזוג בתים מצביעים בעזרת חישוב.

נספח
שורות הקוד בביסיק

HEBREW.BAS

```
10 POKE 106,149:GRAPHICS 0
20 DIM HEB$(216)
30 HEB$="♥♥━RfL|♥♥♥←//_|♥♥♥X↑▘▘[|♥♥♥<▘▘
|f♥♥♥|L_FLXee♥|/_/_|♥♥♥nf66||♥♥♥|↑▘♥♥♥♥♥←
_////"
40 HEB$(73)="/♥|ffff♥♥@|//_<♥♥♥|L_/////
♥Fn8▘[|♥♥♥\vffn♥♥♥◀fff◀♥♥♥|ff_||♥♥♥|nffF
♥♥♥|L_///♥"
50 HEB$(145)="♥♥[|↑▘▘▘♥♥♥FV^LF♥♥♥<↑▘▘▘♥
♥♥|/Ffff♥♥♥Nflx♦♦♦♥[|fflx♥♥♥^VFF|♥♥♥|p00
0♥♥♥|nf///"
100 READ CODE
110 IF CODE=-1 THEN 200
120 POKE 1536+C,CODE
130 C=C+1:GOTO 100
200 A=ADR(HEB$)
210 U=USR(1536,A)
220 POKE 53277,3:POKE 623,1
230 POKE 704,146:POKE 705,146:POKE 706
,146:POKE 707,146
300 RUN "D:SEDER.BAS"
1000 DATA 160,0,104,104,133,219,104,13
3,218,162,0,189,0,224,157,0,156,189,0
1010 DATA 225,157,0,157,189,0,226,157,
0,158,189,0,227,157,0,159,224,255,240,
4,232,76,11,6,200
1020 DATA 192,216,240,8,177,218,153,7,
159,76,43,6,96,-1
```

SEDER.BAS

```
10 C=0:RESTORE 1640
20 READ CODE:IF CODE=-1 THEN 50
30 POKE 38250+C,CODE
40 C=C+1:GOTO 20
50 PMEM=152:POKE 54279,PMEM
60 PMBASE=PMEM*256
70 FOR PARTS=0 TO 3
80 R=0:RESTORE 1600+PARTS*10
90 READ CODE:IF CODE=-1 THEN 120
100 POKE PMBASE+512+(128*PARTS)+57+R,
CODE
110 R=R+1:GOTO 90
120 NEXT PARTS
130 DIM F1$(1):A1=ADR(F1$)
140 A2=INT(A1/1024)+1:A3=(A2*1024-A1)-1
150 DIM F2$(A3)
160 DIM A$(2120)
170 A$="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
180 A$(LEN(A$)+1)="♥♥♥♥♥♥♥┐@♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥%P♥♥♥"
190 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥%T♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
200 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥ %U♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
210 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥◣%U@♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥*%"
220 A$(LEN(A$)+1)="UP♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥*%UT♥♥♥♥♥♥♥♥♥♥"
230 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥ %eUU♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
240 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥◣*%UU@♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
250 A$(LEN(A$)+1)="♥**%Z▀P♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥**%UUT♥♥♥♥♥"
260 A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥ **%UUU♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
```

```
270  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥◣▓▓▓$Q
UU@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
280  A$(LEN(A$)+1)="♥♥♥♥*▓▓▓$QUUP♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▓▓▓▓$QUUT"
290  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥ ▓▓(◣$Q@UU♥♥♥♥♥♥♥♥♥♥♥♥♥"
300  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥◣▓
▓◥$QEUU@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
310  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥*▓▓▓◥$QEUUP♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▓▓▓▓◥$"
320  A$(LEN(A$)+1)="QEUUT♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓▓◥$QEUUU♥♥♥♥♥♥♥♥"
330  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥◣▓▓( ▌◥$QE♥UU@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
340  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥*▓▓▓▓▖Z$Q ▶┓UU
P♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▓▓▓"
350  A$(LEN(A$)+1)="▓▓▖U_▐▶▶┓ UUT♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓▓▓ U_▐▶▓┓ UUU♥♥"
360  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥◣▓▓▓▓▓ U_▐▶▓┓ UUU@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
370  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥*▓▓▓▓ ┣Z▓▓ j
▐▶┣UUUP♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
380  A$(LEN(A$)+1)="♥▓▓▓▓▓▓)V▓k▶UUUUT♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓▓▓▓)V▓k▶UU"
390  A$(LEN(A$)+1)="UUU♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥◣▓▓▓▓▓ ♥U_▐▶▐♥▄UUU@♥♥♥♥♥♥♥♥♥"
400  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥*▓▓▓▓▓
▓U_▐▶▐UUUUUP♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
410  A$(LEN(A$)+1)="♥♥♥♥▓▓▓▓▓▓▓(U j▓▐UUUUUT♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓▓▓▓♥U j"
420  A$(LEN(A$)+1)="▓▐┣UUUUU♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥◣▓▓▓▓▓▓ ♥U_▓▐♥▄UUUU@♥♥♥"
430  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥*▓
▓▓▓▓▓(U_▐▶▐UUUUUUP♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
440  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥▓▓▓▓▓▓▓▓▓┓ _▶▓▄UU
UUUT♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓▓▓▓"
450  A$(LEN(A$)+1)="▓▓ ┣_▐♥▄UUUUUU♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥◣▓▓▓▓▓▓▓▓▓▓(▐QUUUUUUU"
460  A$(LEN(A$)+1)="U@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥*▓▓▓▓▓▓▓▓((▐ QAUUUUUUUP♥♥♥♥♥♥♥♥♥♥♥"
470  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥▓▓▓▓▓▓▓▓▓▀ (▐Q
OUUUUUUUUT♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥ ▓▓▓"
480  A$(LEN(A$)+1)="▓▓▓▓▓▓▓▀ (▐QOUUUUUUUUU♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥◣▓▓▓▓▓▓▓▓▓▀ (▐QOUU"
490  A$(LEN(A$)+1)="UUUUUU@♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥*▓▓▓▓▓▓▓▓▓▓▀ (▐QOUUUUUUUUUP♥♥♥♥♥"
500  A$(LEN(A$)+1)="♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▓▓▓▓▓▓▓▓▓▓
▀UZ▓▶UUUUUUUUUT♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
510  A$(LEN(A$)+1)=" ▓▓▓▓▓▓▓▓▓▓▓/UZ▓▶UUUUUUU
UU♥♥♥♥♥♥♥♥♥♥▶▶▶▶▶▶▶▶▶z▓▓▓▓▓▓▓▓▓▓+▓▶"
520  A$(LEN(A$)+1)="k▶▓UUUUUUUUU ▶▶▶▶▶▶▶▶▶▶▶
▶▶▶▶▶▶▶▶j▓▓▓▓▓▓▓▓▓▓▓+▓▶+▶▓UUUUUUUUU_"
```

```
530 A$(LEN(A$)+1)="▶▶▶▶▶▶▶▶*************
*****?◣◀*?▶UUUUUUUUUW▶▶▶▶▶▶▶▶****"
540 A$(LEN(A$)+1)="************+W] z*:k▶▶
▶▶▶▶▶▶▶▶▶▶▶▶▶▶*****************"
550 A$(LEN(A$)+1)="+W] j*:k▶▶▶▶▶▶▶▶▶▶▶▶▶
▶▶▶****************+W] j*:k▶▶▶▶▶"
560 A$(LEN(A$)+1)="▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶
▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶"
570 A$(LEN(A$)+1)="▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶
▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶"
580 B=ADR(A$):D89=INT(B/256):D88=
B-(D89*256)
590 B=156*256:C=B+64*8:RESTORE 1060
600 FOR M=B TO B+8:POKE M,0:NEXT M
610 READ CODE:IF CODE=-1 THEN 640
620 POKE C,CODE
630 C=C+1:GOTO 610
640 DLIST=38265
650 GRAPHICS 2:SETCOLOR 4,9,10:SETCOLOR
2,1,14:SETCOLOR 0,2,12:SETCOLOR 1,2,6:
POKE 756,159:POKE 559,46
660 POKE 512,106:POKE 513,149
670 POKE 54286,192
680 RESTORE 760
690 FOR C=0 TO 80
700 READ CODE
710 POKE DLIST+C,CODE
720 NEXT C
730 POKE DLIST+12,D88:POKE DLIST+13,D89
740 POKE 561,149:POKE 560,121
750 POKE 53248,112:POKE 53249,120:POKE
53250,128:POKE 53251,136
760 DATA 112,112,112,71,112,147,71,132,
147,6,7,77,0,0,13,13,13,13,13,13,13,
13,13,13
770 DATA 13,13,13,13,13,13,13,13,13,
13,13,13,13,13,13,13,13,13,13,13,13,
13,13,13,13,13,13,13,13,13
780 DATA 13,13,13,141,13,13,13,13,13,
13,66,96,148,2,2,2,65,121,149
790 ? #6;"מכות מצרים"
800 ? #6:? #6;"נדב שטכמן פולישוק"
```

```
840 B=150*256:C=B:RESTORE 1320
850 READ CODE:IF CODE=1 THEN 880
860 POKE C,CODE
870 C=C+1:GOTO 850
880 REM READ PAGE 6 PROGS
890 C=0:RESTORE 1150
900 READ CODE:IF CODE=1 THEN 930
910 POKE 1536+C,CODE
920 C=C+1:GOTO 900
930 REM MANIPULATE KILLER-STARTER RTN
980 POKE 38598+18,PEEK(548):POKE
38598+23,PEEK(549)
990 ?:? "  המשחק להתחלת בג'ויסטיק ירה█←";
1000 IF PEEK(644)=0 THEN 1020
1010 GOTO 1000
1020 GRAPHICS 0
1030 POKE 53277,0
1040 RUN "D:GAME.BAS"
1050 REM CHARACTER DATA (PHARONS)
1060 DATA 0,0,0,0,0,0,0,0
1070 DATA 0,3,15,23,254,90,254,86
1080 DATA 160,103,107,119,170,34,170,138
1090 DATA 0,0,192,80,252,148,252,84
1100 DATA 255,85,47,37,47,37,42,0
1110 DATA 171,221,155,153,155,169,170,0
1120 DATA 252,84,224,96,224,96,160,0
1130 DATA -1
1140 REM VBI MAIN 1536 (MOVING LINES)
1150 DATA 8, 72, 138, 72, 152, 72, 165, 20,
237, 250, 6, 201, 2, 144, 3, 76, 138, 6, 165
1160 DATA 20, 109, 251, 6, 141, 250, 6, 172,
0, 208, 192, 0, 208, 84, 172, 252, 6, 192, 200,
240, 42, 192, 50, 240, 3
1170 DATA 76, 138, 6, 206, 254, 6, 162, 0,
232, 232, 232, 232, 189, 85, 144, 24, 105, 1,
157, 85, 144, 232, 189, 85, 144
1180 DATA 105, 0, 157, 85, 144, 24, 224, 59,
176, 59, 76, 54, 6, 238, 254, 6, 162, 0, 232,
232, 232, 232, 189, 85, 144
```

```
1190 DATA 56,233,1,157,85,144,232,189,
85,144,233,0,157,85,144,24,224,59,176,
24,76,89,6,160,1
1200 DATA 140,30,208,238,253,6,173,252,
6,73,200,73,50,141,4,208,141,252,6,
104,168,104,170,104,40
1210 DATA 76,98,228,170,104,40,76,98,
228,169,50,76,138,6,0,0
1220 REM LINE DOWN 1696
1230 DATA 104,162,56,189,85,144,157,88,
144,202,189,85,144,157,88,144,202,202,
224
1240 DATA 3,144,3,76,163,6,169,152,
157,88,144,202,169,20,157,88,144,96
1250 REM MOVEPM U(1733) 1743,8 1748,0
1260 DATA 104,104,104,141,2,208,141,0,
208,105,0,141,3,208,105,8,141,1,208,96
1270 REM DLI 0-1755 (1-705) 2-1760 3-1765
1280 DATA 72,169,236,141,10,212,141,18,
208,169,54,141,20,208,169,212,141,21,208
1290 DATA 169,16,141,27,208,104,64,-1
1300 REM MUSIC GENRATOR
1310 REM POKE 39457,8 - MUSIC TEMPO
1320 DATA 104,169,11,141,40,2,169,150,
141,41,2,238,255,6,174,255,6,189,0,
151,201,254
1330 DATA 176,14,141,0,210,169,170,141,
1,210,169,16,141,26,2,96,162,0,142,
255,6,169,0,76,24,150,0,0
1340 REM CLEAN 38450
1350 DATA 162,94,104,104,104,201,1,240,
7,201,2,240,19,76,98,150,189,128,153
1360 DATA 9,48,157,128,153,202,224,3,
144,19,76,66,150,189,128,153,41,207,
157,128,153,202,224,3,144
1370 DATA 3,76,82,150,96,0,0
1390 REM SHOOT 38501
1400 DATA 104,104,104,168,104,104,170,
192,48,240,7,192,12,240,41,76,231,150,
189,144
```

```
1410 DATA 153,41,207,157,144,153,232,
189,144,153,41,207,157,144,153,232,
189,144,153,41,207,157,144,153,232
1420 DATA 189,144,153,9,48,157,144,153,
76,231,150,189,128,153,9,12,157,128,
153,232,189,128,153,41,243,157
1430 DATA 128,153,232,189,128,153,41,
243,157,128,153,232,189,128,153,41,
243,157,128,153,76,231,150,96,0,0
1435 REM STARTER-KILLER 38598
1440 DATA 104,104,104,166,20,232,228,20,
208,3,76,204,150,201,1,240,13,169,85
1450 DATA 141,36,2,169,192,141,37,2,
76,238,150,169,0,141,36,2,169,6,141,
37,2,96,0,0
1455 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1460 REM MUSIC NOTES
1470 DATA 0,0,0,0,108,108,0,108,108,0,
108,108,0,108,108,0,108,108,0,81,81,
81,81,0,0,96,96,0,91,91,91,91,91,0
1480 DATA 96,96,96,96,96,0,108,108,
108,108,0,0,0,0,0,144,144,0,108,108,0
1490 DATA 108,108,0,108,108,0,108,108,
0,108,108,0,81,81,81,81,0,0,96,96,0
1500 DATA 91,91,91,91,91,0,96,96,96,
96,96,0,108,108,108,108,0,0,0,0
1510 DATA 0,0,0,72,72,72,72,0,0,72,72,
72,72,0,0,60,60,0,72,72,72,72,0,0,0,0
1520 DATA 72,72,72,72,0,0,81,81,81,81,
0,0,72,72,0,108,108,108,108,0,0
1530 DATA 108,108,0,81,81,0,81,81,0,81,
81,0,81,81,0,81,81,0,68,68,0,72,72,0
1540 DATA 81,81,0,91,91,91,91,91,0,98,
98,98,98,98,0,108,108,108,108,0,0,0,0
1550 DATA 0,108,108,0,81,81,0,81,81,0,
81,81,0,81,81,0,81,81,0,68,68,0,72,72
1560 DATA 0,81,81,0,91,91,91,91,91,0,
98,98,98,98,98,0,108,108,108,108,0,0,
0,255,-1
```

```
1570 REM PYRAMID DATA
1600 DATA 0,0,3,6,7,7,15,15,1,56,56,
57,57,112,126,126,64,126,62,48,0,62,
28,28,7,6,0,0,0,0,-1
1610 DATA 54,54,246,54,182,182,182,182,
182,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
236,108,108,108,108,108,-1
1620 DATA 216,216,223,216,219,219,219,
219,219,216,0,1,1,0,0,0,0,0,0,0,0,0,
0,0,223,217,216,216,216,216,-1
1630 DATA 0,0,128,192,192,192,224,224,
0,56,56,56,56,28,252,252,4,252,248,
24,0,248,112,112,128,128,0,0,0,0,-1
1635 REM LOCAL DLI
1640 DATA 72,169,30,234,234,234,141,26,
208,104,64,-1
```

GAME.BAS

```
10 DIM MAKA$(10):DIM HIT(7):MAKOT=1:
POKE 1785,0:REM 1785 - INFO FLAG
20 REM MEMORY MAP: 149=NOTES; 150=HIGHMEM
ROUTINE; 152=PM; 156=HEBREW+CHAR
30 POKE 77,0:REM PREVENTING ATTRACT MODE
40 IF MAKOT>10 THEN POKE 559,0:
RUN "D:WINNER.BAS"
45 ROW1=7:ROW2=7:ROW3=7:PMX=1733:
PMY=38501:DLI=1753:KILLDLI=72
50 PHAR=21:CHK=1:CHKLINE=22:GOVL=28:
PHARY=1790:CLN=38450:NAGAN=38400:
STAKIL=38598
60 POKE 1612,59:POKE 1647,59:POKE 1698,56:
REM RESETING DOWN.ASM AND MAINVBI.ASM
70 POKE 1788,200:POKE 1789,0:POKE PHARY,100:
POKE 1791,0:LINE=2:V=45:YTIL=91:TIL=0:SHURA=1
80 POKE 1787,19-MAKOT
90 REM MAKAT-MIZRAYM
100 PMEM=152:POKE 54279,PMEM
110 PMBASE=PMEM*256:FOR R=PMBASE+384 TO
PMBASE+478:POKE R,243:NEXT R
120 FOR R=PMBASE+479 TO PMBASE+1024:
POKE R,0:NEXT R
130 FOR MAKA=0 TO 3
140 R=0:RESTORE 1720+((MAKOT-1)*60)+MAKA*10
150 READ CODE:IF CODE=-1 THEN 180
160 POKE PMBASE+512+128*MAKA+101+R,CODE
170 R=R+1:GOTO 150
180 NEXT MAKA
190 POKE 53277,3:POKE 53252,200:POKE 623,8
200 READ CODE:POKE 53258,CODE:READ CODE:
POKE 53259,CODE
210 READ PL1:POKE PMX+10,PL1:READ PL2:
POKE PMX+15,PL2
220 READ COL0:POKE DLI+2,COL0:READ COL1:
POKE 705,COL1
230 READ COL2:POKE DLI+10,COL2:READ COL3:
POKE DLI+15,COL3
```

```
240 READ COL10:READ COL11:READ COL12:
READ MAKA$
250 REM GRAPHICS 2 COMPLIMENT
260 GRAPHICS 2+16:POKE 559,0:POKE
756,156+3:SETCOLOR 2,0,0:SETCOLOR
0,9,12:SETCOLOR 1,2,14
270 POKE 704,0:POKE 706,0:POKE 711,15
280 DLIST2=PEEK(561)*256+PEEK(560):POKE
DLIST2+15,PEEK(DLIST2+15)+128
285 POKE DLIST2+11,2:POKE DLIST2+12,2
287 VV=PEEK(89)*256+PEEK(88)+121
290 POSITION 14-LEN(MAKA$),4:? #6;MAKA$
291 IF PEEK(1785) THEN 300
292 OPEN #1,4,0,"D:INFO.TXT"
293 FOR C=0 TO (MAKOT-1)*77:GET #1,CODE:
NEXT C
294 FOR C=0 TO 75:GET #1,CODE:POKE
VV+C,CODE:NEXT C
295 CLOSE #1
300 U=USR(PMX,120)
310 POKE 512,217:POKE 513,6:POKE 54286,192
320 POKE 559,46
330 FOR TIME=1 TO 2500+10000*( NOT
(PEEK(1785))):NEXT TIME:U=USR(PMX,0)
340 GRAPHICS 12+16:POKE 559,0:POKE 87,14:
POKE 756,156+3
350 SETCOLOR 2,2,8:SETCOLOR 0,8,10:
SETCOLOR 1,1,14:SETCOLOR 4,0,0:POKE 711,15
360 REM DISPLAY LIST MAKING
370 DLM=PEEK(89):DLL=PEEK(88):
MEM=DLM*256+DLL
380 RESTORE 1680
390 FOR X=MEM-75 TO MEM:READ CODE:
POKE X,CODE:NEXT X
400 DLIST=MEM-75:POKE 561,DLM:POKE
560,DLL-75
410 REM PHARONS LINES
420 FOR Z=1 TO 56:POKE MEM+Z,0:NEXT Z
430 FOR W=0 TO 6 STEP 3:FOR X=0 TO 1:
FOR Y=0 TO 30 STEP 5
```

```
440 FOR Z=1 TO 3:
POKE 40+MEM+40*(W+X)+Z+Y-1,64+Z+(X*3)
450 NEXT Z:NEXT Y:NEXT X:NEXT W
460 REM DROWING PYRAMIDS
470 FOR M=0 TO 14 STEP 7
480 FOR N=0 TO 2 STEP 2
490 POKE MEM+701+N+M,1:POKE MEM+702+N+M,128
500 POKE MEM+721+N+M,3:POKE MEM+722+N+M,192
510 POKE MEM+741+N+M,7:POKE MEM+742+N+M,224
520 POKE MEM+761+N+M,15:POKE MEM+762+N+M,240
530 POKE MEM+781+N+M,31:POKE MEM+782+N+M,248
540 POKE MEM+801+N+M,63:POKE MEM+802+N+M,252
550 POKE MEM+821+N+M,127:POKE MEM+822+N+M,254
560 POKE MEM+841+N+M,255:POKE MEM+842+N+M,255
570 NEXT N:NEXT M
580 POKE 512,217:POKE 513,6:POKE 54286,192
590 POKE 20,0:POKE 1786,40
600 POKE 53278,1:U=USR(STAKIL,1)
610 U=USR(PMX,45)
620 POKE 559,46
630 POKE NAGAN+33,9-INT(MAKOT/2):U=USR(NAGAN)
640 REM MAIN LOOP
650 IF PHAR=0 THEN 1350
660 IF ROW3<1 THEN CHKLINE=28*CHK:
GOVL=34*(NOT CHK)
670 IF (ROW3<1) AND (ROW2<1) THEN
CHKLINE=34*CHK:GOVL=40*(NOT CHK)
680 IF PEEK(1789)=LINE THEN
LINE=LINE+2:A=USR(1696)
690 IF PEEK(1789)=CHKLINE THEN GOSUB 1280
700 IF PEEK(1789)=GOVL THEN 1490
710 L1=15+(LINE*2):L3=L1+24
720 IF PEEK(632)=7 THEN U=USR(PMX,V+2):
V=V+2
730 IF PEEK(632)=11 THEN U=USR(PMX,V-2):
V=V-2
740 IF V>190 THEN V=190
750 IF V<46 THEN V=45
760 IF (NOT POY1) THEN
POY1=((INT(RND(0)*(84-(MAKOT*8))))=2)
```

```
780 IF POY1=1 THEN XOY1=V+7:POKE
53254,XOY1
790 IF (POY1=1) AND (PEEK(53250)<>0)
THEN POY1=2:YOY1=LINE*2:POKE 53278,1:
U=USR(CLN,2)
800 REM (2)=KIBUI #2
810 IF POY1=2 THEN GOSUB 850
820 IF (TIL=0) AND (PEEK(644)=0)
THEN TIL=1:XTIL=V+7
830 IF TIL=1 THEN GOSUB 990
840 GOTO 640
850 REM OIEV ON AIR
860 IF YOY1>71 AND YOY1<75 AND CHK=1
THEN FOR R=36 TO 44:LOCATE XOY1-48,R,FF:
MM=MM+FF:NEXT R
870 IF MM>0 THEN U=USR(PMY,48,YOY1):
YOY1=YOY1+3:GOSUB 930
880 IF PEEK(53258)<>0 THEN 1490
890 IF YOY1>98 THEN POKE
706,0:POY1=0:POKE (PMBASE+400+YOY1),0:
U=USR(CLN,1):POKE 53254,0:POKE 53278,1:
RETURN
900 IF PEEK(706)=0 THEN IF PEEK(53250)<>0
OR YOY1-10>LINE*2 THEN POKE 706,15
910 U=USR(PMY,48,YOY1):YOY1=YOY1+3
920 RETURN
930 REM PYRAMID EROSE
960 COLOR 0:PLOT XOY1-48,35:
DRAWTO XOY1-48,38+5*(MM<5)
970 PLOT XOY1-49,35:DRAWTO XOY1-49,
38+5*(MM<5):PLOT XOY1-47,35:DRAWTO XOY1-47,
38+5*(MM<5)
980 MM=0:POKE 53254,0:RETURN
990 REM TIL ON AIR
1000 POKE 53253,XTIL:R=PMBASE+384+YTIL
1010 U=USR(PMY,12,YTIL):YTIL=YTIL-3
1020 IF YTIL<3 THEN R=PMBASE+387+YTIL:
POKE R,PEEK(R)-12:TIL=0:YTIL=91:RETURN
1030 IF PEEK(53249)<>0 AND (YTIL<88 OR
CHK=0) THEN 1060
```

```
1040 IF CHK=1 AND PEEK(53249)<>0 AND
YTIL>87 THEN POKE 53278,1
1050 RETURN
1060 REM TIL HIT PHARON
1070 SOUND 1,12,4,10
1080 POKE R,3:POKE R+1,3:POKE 53278,1:
TIL=0
1090 X1=XTIL:Y1=YTIL:YTIL=91+10*(NOT
(CHK))
1100 P=4*(PEEK(PHARY)-100):X2=X1-P
1110 HIT(0)=ABS(X2-58):HIT(1)=ABS(X2-75)
1120 HIT(2)=ABS(X2-98):HIT(3)=ABS(X2-115):
HIT(4)=ABS(X2-135)
1130 HIT(5)=ABS(X2-154):HIT(6)=ABS(X2-173)
1140 HITX=0:TEMP=HIT(0)
1150 FOR C=1 TO 6
1160 IF HIT(C)<TEMP THEN TEMP=HIT(C):HITX=C
1165 NEXT C
1170 HITY=2:IF ABS(L1-Y1)<8 THEN HITY=1
1180 IF ABS(L3-Y1)<8 THEN HITY=3
1190 IF HITY=1 THEN
DL1=DLM*256+200:DL2=DL1+40:ROW1=ROW1-1
1200 IF HITY=2 THEN DL1=(DLM+1)*256+64:
DL2=DL1+40:ROW2=ROW2-1
1210 IF HITY=3 THEN ROW3=ROW3-1:
DL1=(DLM+1)*256+184:DL2=DL1+40
1230 IF PEEK(DL1+HITX*5+0)=0 THEN GOSUB
1273
1240 FOR X=0 TO 3:POKE DL1+HITX*5+X,0:
POKE DL2+HITX*5+X,0:NEXT X
1250 PHAR=PHAR-1
1260 SOUND 1,0,0,0
1270 GOTO 840
1271 REM TAUT BAZIHUI
1273 PHAR=PHAR+1
1274 IF HITY=1 THEN ROW1=ROW1+1
1275 IF HITY=2 THEN ROW2=ROW2+1
1276 IF HITY=3 THEN ROW3=ROW3+1
1277 RETURN
```

```
1280 REM CHKLINE
1290 CHK=0:CHKLINE=0:KILLDLI=66
1300 POKE DLIST+60,68:POKE DLIST+61,0:
POKE DLIST+62,143:POKE DLIST+63,68:
POKE DLIST+64,0:POKE DLIST+65,143
1310 POKE DLIST+66,204:POKE DLIST+67,0:
POKE DLIST+68,143
1320 POKE DLIST+69,65:POKE DLIST+70,140:
POKE DLIST+71,85:POKE DLIST+72,0
1330 POKE 1612,65:POKE 1647,65:
POKE 1698,62:REM MANIPULATING DOWN.ASM
AND MAINVBI.ASM
1340 RETURN
1350 REM WIN ROUND
1360 POKE NAGAN+33,0:REM STOP MUSIC
1370 U=USR(STAKIL,0):FOR TIME=1 TO
100:SOUND 0,0,0,0:NEXT TIME
1380 POKE 53252,0:POKE 53253,0:POKE
53254,0:POKE 53255,0
1390 RESTORE 2330
1400 READ CODE
1410 IF CODE=-1 THEN 1460
1420 SOUND 0,CODE,10,10
1430 POKE 712,INT(RND(0)*255)
1440 FOR TIME=1 TO 80:NEXT TIME
1450 GOTO 1400
1460 SOUND 0,0,0,0:POKE 712,0:FOR TIME=1
TO 1000:NEXT TIME
1470 MAKOT=MAKOT+1
1475 U=USR(PMX,0)
1480 POY1=0 POKE 1785,0:GOTO 30
1490 REM GAME OVER
1500 POKE NAGAN+33,0:REM STOP MUSIC
1510 U=USR(STAKIL,0)
1520 POKE DLIST+KILLDLI,
PEEK(DLIST+KILLDLI)-128
1530 POKE 53252,0:POKE 623,16
1540 POKE 704,COL10:POKE 705,COL10
1550 POKE 706,COL11:POKE 707,COL12
```

```
1560 FOR Z=1 TO 240:SOUND 0,Z,6,10:
FOR TIME=1 TO 5:NEXT TIME:NEXT Z
1570 RESTORE 2370
1580 READ CODE
1590 IF CODE=-1 THEN 1630
1600 SOUND 0,CODE,10,10
1610 FOR TIME=1 TO 120:NEXT TIME
1620 GOTO 1580
1630 GRAPHICS 2:SETCOLOR 2,0,0:
POKE 756,156+3
1640 POSITION 0,5:? #6;"ויכבד לב פרעה  "
1650 ? #6;"ולא שלח את העם  "
1660 FOR TIME=1 TO 3600:NEXT TIME:
U=USR(PMX,0):POKE 53254,0:POKE 53253,0
1661 ? "ירה בג'ויסטיק כדי לנסות שוב...  
█";CHR$(30);
1663 IF STRIG(0)=0 THEN 1666
1664 GOTO 1663
1666 POKE 1785,1:GOTO 30
1670 REM DISPLAY LIST
1680 DATA 112,112,112,68,200,144,68,
240,144,68,24,145,68,64,145,68,104,
145,68,144,145,68,184,145,68,224,145
1690 DATA 68,8,146,68,48,146,68,88,
146,68,128,146,68,128,146,68,128,146,
68,128,146,68,127,146
1700 DATA 68,128,146,68,128,146,68,128,
146,68,128,146,76,92,147,12,12,12,12,
12,12,140,65,85,144
1710 REM DAM
1720 DATA 1,2,2,4,4,8,16,16,32,40,40,
36,19,8,7,0,0,0,0,0,-1
1730 DATA 0,128,128,64,64,32,16,16,8,
8,8,8,16,32,192,0,0,0,0,0,-1
1740 DATA 0,1,1,3,3,7,15,15,31,23,23,
27,12,7,0,0,0,0,0,0,-1
1750 DATA 0,0,0,128,128,192,224,224,
240,240,240,240,224,192,0,0,0,0,0,0,-1
1760 DATA 0,0,8,0,50,50,52,52,182,178,
178, דם
```

```
1770 REM ZEFARDEA
1780 DATA 16,50,20,16,4,4,10,28,48,24,
8,4,4,0,0,0,0,0,0,0,-1
1790 DATA 16,152,80,16,0,0,160,112,24,
48,32,64,64,0,0,0,0,0,0,0,-1
1800 DATA 0,5,11,15,11,11,5,3,0,0,0,
0,0,0,0,0,0,0,0,0,-1
1810 DATA 0,64,160,224,224,224,64,128,
0,0,0,0,0,0,0,0,0,0,0,0,-1
1820 DATA 0,0,8,0,198,198,220,220,70,
75,75,צפרדע
1830 REM KINIM
1840 DATA 48,8,4,3,68,36,23,9,121,137,
153,41,41,69,131,0,0,0,0,0,-1
1850 DATA 24,32,64,128,68,72,208,32,
60,34,50,40,40,68,130,0,0,0,0,0,-1
1860 DATA 0,0,0,0,3,3,0,6,6,6,6,6,6,
2,0,0,0,0,0,0,-1
1870 DATA 0,0,0,0,128,128,0,192,192,
192,192,192,192,128,0,0,0,0,0,0,-1
1880 DATA 0,0,8,0,40,40,36,36,8,248,
248,כינים
1890 REM AROV
1900 DATA 31,127,207,192,224,192,192,
199,248,248,120,127,15,3,0,0,0,0,0,0,
-1
1910 DATA 224,248,204,12,28,12,12,140,
124,124,120,248,192,0,0,0,0,0,0,0,-1
1920 DATA 0,0,68,124,124,84,84,108,0,
0,0,0,68,40,16,0,0,0,0,0,-1
1930 DATA 0,0,0,129,0,34,34,0,60,0,0,
0,0,0,0,0,0,0,0,0,-1
1940 DATA 1,0,3,5,40,40,26,15,54,52,0,
ערוב
1950 REM DEVER
1960 DATA 3,17,33,83,15,14,155,254,
155,14,15,83,33,17,3,0,0,0,0,0,-1
1970 DATA 128,16,8,148,224,224,178,254,
178,224,224,148,8,16,128,0,0,0,0,0,-1
1980 DATA 0,0,0,0,0,1,4,1,4,1,0,0,0,
0,0,0,0,0,0,0,-1
```

```
1990 DATA 0,0,0,0,0,0,64,0,64,0,0,0,
0,0,0,0,0,0,0,0,-1
2000 DATA 0,0,8,0,200,200,72,72,6,24,
24,דבר
2010 REM SHKHIN
2020 DATA 1,9,9,13,13,13,77,71,102,
116,126,63,31,4,1,3,3,3,0,0,-1
2030 DATA 32,32,50,178,178,182,246,206,
252,124,228,200,248,240,240,240,176,
16,0,0,-1
2040 DATA 0,0,0,0,0,0,0,0,1,3,1,0,0,
3,2,0,0,0,0,-1
2050 DATA 0,0,0,0,0,0,0,48,0,128,24,
48,0,0,0,0,64,224,0,-1
2060 DATA 0,0,8,0,44,44,54,54,12,42,
42,שכין
2070 REM BARAD
2080 DATA 0,0,13,18,16,32,64,128,128,
255,0,32,80,145,146,98,2,1,0,0,-1
2090 DATA 96,144,8,4,12,6,1,1,2,252,0,
8,20,18,146,76,64,128,0,0,-1
2100 DATA 0,0,0,13,15,31,63,127,127,0,
0,0,32,96,97,1,1,0,0,0,-1
2110 DATA 0,96,240,248,240,248,254,254,
252,0,0,0,8,12,12,128,128,0,0,-1
2120 DATA 0,0,8,0,114,114,10,10,10,68,
68,ברד
2130 REM ARBE
2140 DATA 24,24,4,2,15,31,63,127,227,
193,193,193,227,123,60,31,7,3,1,1,-1
2150 DATA 48,48,64,128,224,240,248,252,
142,6,6,6,142,188,120,240,192,128,0,0,
-1
2160 DATA 0,0,0,0,0,0,0,0,252,254,182,
182,254,-1
2170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,48,126,125,90,181,-1
2180 DATA 1,1,0,8,220,220,38,196,52,0,
252,ארבה
```

```
2190 REM KHOSHEKH
2200 DATA 0,0,0,0,0,0,0,0,16,28,31,15,
7,1,0,0,0,0,0,0,-1
2210 DATA 0,0,0,224,112,120,56,60,124,
252,248,248,240,192,0,0,0,0,0,0,-1
2220 DATA 0,0,0,20,8,20,0,0,0,0,0,0,
0,80,32,80,0,0,0,0,-1
2230 DATA 0,0,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,
255,0,0,-1
2240 DATA 0,1,0,8,14,14,30,4,0,0,236,
חושך
2250 REM BEKHOROT
2260 DATA 15,31,63,63,59,49,179,158,
71,5,0,10,15,39,192,64,0,0,0,0,-1
2270 DATA 224,240,248,248,184,24,154,
242,196,64,0,160,224,200,6,4,0,0,0,0,-1
2280 DATA 0,0,0,0,4,14,12,1,8,10,15,5,
0,0,0,0,0,0,0,0,-1
2290 DATA 0,0,0,0,64,224,96,0,32,160,
224,64,0,0,0,0,0,0,0,0,-1
2300 DATA 0,0,8,0,6,6,15,15,54,0,0,
מכת בכורות
2320 REM WINNING MUSIC
2330 DATA 121,121,0,91,91,0,91,91,0,81,
81,0,96,96,0,91,91,0,96,108,108,121,121
2340 DATA 0,121,121,0,108,108,0,108,
108,0,96,96,0,121,121,0,91,91,0,91,
91,0,91,91,91
2350 DATA -1
2360 REM GAME OVER MUSIC
2370 DATA 108,108,108,108,108,0,108,
108,0,108,108,0,121,121,121,136,136,
136,144,144,144,136,136,136,121,121,121
2380 DATA 121,0,0,108,108,108,121,121,
121,136,136,136,144,144,144,162,162,
162,0,0,0,173,173,173,173,173,0
2390 DATA 162,162,162,144,144,144,108,
108,108,108,0,0,217,217,217,217,0,0,
162,162,162,162,162,162,0
2400 DATA -1
```

WINNER.BAS

```
10 POKE 53248,0:POKE 53249,0:POKE 53250,0:
POKE 53251,0:POKE 1791,0
20 DIM F1$(1):A1=ADR(F1$)
30 A2=INT(A1/1024)+1:A3=(A2*1024-A1)-1
40 DIM F2$(A3)
50 DIM A$(720)
60 A$="yy♥[[[[♥♥[[[QQQHHHQQQ[[[♥♥♥QQQ[[[
♦♦♦111♦♦♦111yyy♥♥♥[[[[♥♥[[[QQQHHHQQQ[[[♥
♥♥QQQ[["
70 A$(81)="[♦♦♥1♦♥[[[♥♥♥♥♥♥yy♥[[[[♥♥[[[Q
QQHHHQQQ[[[♥♥♥QQQ[[[♦♦♦111♦♦♦111yyy♥♥♥[[
[[♥♥[[[QQQH"
80 A$(161)="HHQQQ[[[♥♥♥QQQ[[[♦♦♥1♦♥[[[♥♥
♥♥♥♥♥HH♥<<<<<♥♥♥♥55♥<<<<DDHHHQQHHHHH♥QQ
Q[[[QQQ♥♥♥♥♥"
90 A$(241)="♥♥♥<<<<<♥♥♥♥55":A$(255,255)=
CHR$(255)
100 A$(LEN(A$)+1)="5♥<<<<DDHHHQQHHHHH♥Q
QQ[[[QQQ♥♥♥♥♥♥QQQHHHHHH<<<<<<QQQQQQ♥♥♥[[
[♦♦♥[["
110 A$(LEN(A$)+1)="[QQQHHHDDD♥♥♥♥♥♥♥HHH
HH♥<<<<<♥QQQQQ♥♥♥HHHQQQ[[[♦♦♦[[[QQQ♥♥♥♥♥
♥♥"
120 A$(LEN(A$)+1)=CHR$(255)
130 AD=ADR(A$)
140 C=0:RESTORE 750
150 READ CODE:IF CODE=1 THEN 180
160 POKE 1536+C,CODE
170 C=C+1:GOTO 150
180 U=USR(1536,AD,1)
190 U=USR(1536,AD,2)
200 U=USR(1670)
210 A$(1)=CHR$(0):A$(720)=CHR$(0)
220 A$(2)=A$
230 A$(13)="AB♥♥♥AB♥♥♥AB♥♥♥AB♥♥♥AB♥♥♥AB"
240 A$(53)="EF♥♥♥EF♥♥♥EF♥♥♥EF♥♥♥EF♥♥♥EF"
250 A$(93)="NO♥♥♥NO♥♥♥NO♥♥♥NO♥♥♥NO♥♥♥NO"
260 A$(13+120)="CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥
♥CD"
270 A$(53+120)="GH♥♥♥GH♥♥♥GH♥♥♥GH♥♥♥GH♥♥
♥GH"
280 A$(93+120)="PQ♥♥♥PQ♥♥♥PQ♥♥♥PQ♥♥♥PQ♥♥
♥PQ"
```

```
290 A$(13+240)="CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥
♥CD"
300 A$(53+240)="IJ♥♥♥IJ♥♥♥IJ♥♥♥IJ♥♥♥IJ♥♥
♥IJ"
310 A$(93+240)="RS♥♥♥RS♥♥♥RS♥♥♥RS♥♥♥RS♥♥
♥RS"
320 A$(13+360)="AB♥♥♥AB♥♥♥AB♥♥♥AB♥♥♥AB♥♥
♥AB"
330 A$(53+360)="KLZ♥♥KLZ♥♥KLZ♥♥KLZ♥♥KLZ♥
♥KLZ"
340 A$(93+360)="TU♥♥♥TU♥♥♥TU♥♥♥TU♥♥♥TU♥♥
♥TU"
350 A$(13+480)="CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥
♥CD"
360 A$(53+480)="MJ♥♥♥MJ♥♥♥MJ♥♥♥MJ♥♥♥MJ♥♥
♥MJ"
370 A$(93+480)="VW♥♥♥VW♥♥♥VW♥♥♥VW♥♥♥VW♥♥
♥VW"
380 A$(13+600)="CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥♥CD♥♥
♥CD"
390 A$(53+600)="KJ♥♥♥KJ♥♥♥KJ♥♥♥KJ♥♥♥KJ♥♥
♥KJ"
400 A$(93+600)="XY♥♥♥XY♥♥♥XY♥♥♥XY♥♥♥XY♥♥
♥XY"
410 PMEM=152:POKE 54279,PMEM
420 PMBASE=PMEM*256
430 FOR PARTS=0 TO 3
440 R=0:RESTORE 1140+PARTS*10
450 READ CODE:IF CODE=-1 THEN 480
460 POKE PMBASE+512+(128*PARTS)+81+R,CODE
470 R=R+1:GOTO 450
480 NEXT PARTS
490 B=156*256:C=B+65*8:RESTORE 850
510 READ CODE:IF CODE=-1 THEN 550
520 POKE C,CODE
530 C=C+1:GOTO 510
550 GRAPHICS 2:POKE 559,0:SETCOLOR
0,8,3:SETCOLOR 1,2,10:SETCOLOR
4,9,10:SETCOLOR 2,1,14:POKE 756,156+3
560 POKE 704,36:POKE 705,36:POKE 706,28:
POKE 707,28
570 POKE 53248,47:POKE 53249,47+8:POKE
53250,47+16:POKE 53251,47+24:POKE 559,46
580 U=USR(1590)
590 AD=ADR(A$): DLIST=PEEK(561)*256+
PEEK(560)
```

```
600 POKE DLIST+12,69:POKE DLIST+15,5:
POKE DLIST+16,5:POKE DLIST+17,66:POKE
DLIST+18,96:POKE DLIST+19,148
610 POKE DLIST+3,7+128+64
620 POKE 54286,192:POKE 512,161:POKE 513,6
630 POKE 559,46
640 ? #6:? #6;"   בצאת ישראל ממצרים":? #
6;"   בית יעקב מעם לועז"
650 LOOP=LOOP+1:FOR CASE=0 TO 600 STEP 120
660 TM=AD+CASE:U=USR(1658,TM)
670 FOR TIME=1 TO 42
680 IF OPT=2 AND STRIG(0)=0 THEN OPT=3
690 NEXT TIME
700 NEXT CASE
710 IF LOOP=10 THEN OPT=1
720 IF OPT=1 THEN OPT=2:? :? "   לבייסיק
 ליציאה בגיויסטיק ירה█";CHR$(30);
730 IF OPT=3 THEN POKE 1791,7:FOR
C=53248 TO 53251:POKE C,0:NEXT C:POKE
1630,0:POKE 106,160:GRAPHICS 0:END
740 GOTO 650
749 REM PAGE 6 PROGRAMS
750 DATA 160, 0, 104, 104, 133, 204, 104,
133, 203, 104, 104, 201, 1, 240, 7, 201, 2,
240, 16, 76, 50, 6, 177, 203, 153, 0, 150, 201
760 DATA 254, 176, 19, 200, 76, 22, 6, 230,
204, 177, 203, 153, 0, 151, 201, 254, 176, 4,
200, 76, 37, 6, 96, 0, 0, 0
770 DATA 104, 169, 0, 133, 203, 169, 150, 133,
204, 169, 73, 141, 40, 2, 169, 6, 141, 41, 2
780 DATA 238, 255, 6, 172, 255, 6, 177, 203,
201, 254, 176, 16, 141, 0, 210, 169, 170,
141, 1, 210, 169, 6,141, 26, 2
790 DATA 76, 119, 6, 165, 204, 73, 150, 73,
151, 133, 204, 162, 0, 142, 255, 6, 169, 0,
76, 85, 6, 96, 0, 0
800 DATA 104, 104, 170, 104, 141, 101, 147,
142, 102, 147, 96, 0
810 DATA 104, 169, 0, 162, 0, 157, 0, 152,
157, 0, 153, 157, 0, 154, 157, 0, 155, 232, 224
820 DATA 254, 176, 3, 76, 139, 6, 96, 0
```

```
830 DATA 72, 138, 72, 152, 72, 166, 20, 160,
50, 142, 10, 212, 142, 22, 208, 202, 136, 208,
246
840 DATA 140, 22, 208, 104, 168, 104, 170,
104, 64, 0, -1
849 REM CHARACTER DATA (WALKING MAN)
850 DATA 0,1,6,6,10,2,0,2
860 DATA 0,84,168,100,168,168,128,168
870 DATA 1,6,6,10,2,0,10,42
880 DATA 84,168,100,168,168,128,168,168
890 DATA 10,9,9,9,9,9,9,2
900 DATA 168,164,88,168,164,164,88,172
910 DATA 38,37,38,38,38,37,10,14
920 DATA 152,100,168,152,152,105,189,188
930 DATA 38,37,38,38,38,37,42,10
940 DATA 152,100,168,152,152,169,253,252
950 DATA 42,38,37,166,166,38,37,43
960 DATA 168,153,101,169,153,153,169,253
970 DATA 38,37,166,166,38,37,43,43
980 DATA 15,15,15,15,15,32,128,32
990 DATA 172,252,252,200,8,8,8,10
1000 DATA 15,15,15,15,0,10,8,0
1010 DATA 252,252,200,200,136,8,32,40
1020 DATA 15,15,15,15,0,2,2,0
1030 DATA 252,252,232,232,32,160,128,160
1040 DATA 43,15,15,15,15,2,8,2
1050 DATA 252,252,252,188,40,136,8,10
1060 DATA 15,15,15,15,0,2,2,2
1070 DATA 252,252,188,136,136,160,0,128
1080 DATA 15,15,15,15,0,2,8,10
1090 DATA 252,252,188,130,138,32,40,0
1100 DATA 0, 0, 0, 0, 0, 0, 64, 64, -1
1120 REM PYRAMID DATA (PM)
1140 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 7,
15, 31, 63, -1
1150 DATA 0, 1, 3, 7, 15, 31, 63, 127, 255,
255, 255, 255, 255, 255, 255, -1
1160 DATA 192, 224, 240, 248, 252, 254, 255,
255, 255, 255, 255, 255, 255, 255, 255, -1
1170 DATA 0, 0, 0, 0, 0, 0, 0, 128, 192, 224,
240, 248, 252, 254, 255, -1
```

ביחידת לימוד זו אספר ביציאת מצרים ואסביר את שורות הקוד אשר בעזרתן מתרוצצות על המסך עשר מכות מצרים ומשמידות את חיל פרעה.

אציג את הטכניקות בהן כתבתי את משחק המחשב ואת שילוב הכוחות בין הפעלת גרפיקת השחקנים, שהוצגה במחשבת 7; שימוש מתקדם בפקודות אסמבלי, שנלמדו במחשבת 8; ורתימתן של הפרעות ריענון המסך (שהוצגו במחשבת 9) לצרכים תכנותיים.

נכתב ע"י: נדב שטכמן פולישוק